Secure Connections Overview

A virtual private network (VPN) connection establishes a secure tunnel between endpoints over a public network such as the Internet.

This chapter applies to Remote Access and Site-to-site VPNs on Secure Firewall Threat Defense devices. It describes the Internet Protocol Security (IPsec), the Internet Security Association and Key Management Protocol (ISAKMP, or IKE) and SSL standards that are used to build site-to-site and remote access VPNs.

VPN Types

The Firewall Management Center supports the following types of VPN connections:

- Remote Access VPNs on Firewall Threat Defense devices.

  Remote access VPNs are secure, encrypted connections, or tunnels, between remote users and your company's private network. The connection consists of a VPN endpoint device, which is a workstation or mobile device with VPN client capabilities, and a VPN headend device, or secure gateway, at the edge of the corporate private network.

  Secure Firewall Threat Defense devices can be configured to support Remote Access VPNs over SSL or IPsec IKEv2 by the Firewall Management Center. Functioning as secure gateways in this capacity, they authenticate remote users, authorize access, and encrypt data to provide secure connections to your network. No other types of appliances, managed by the Firewall Management Center, support Remote Access VPN connections.

  Secure Firewall Threat Defense secure gateways support the Secure Client full tunnel client. This client is required to provide secure SSL IPsec IKEv2 connections for remote users. This client gives remote users the benefits of a client without the need for network administrators to install and configure clients

on remote computers since it can be deployed to the client platform upon connectivity. It is the only client supported on endpoint devices.

- Site-to-site VPNs on Firewall Threat Defense devices.

  A site-to-site VPN connects networks in different geographic locations. You can create site-to-site IPsec connections between managed devices, and between managed devices and other Cisco or third-party peers that comply with all relevant standards. These peers can have any mix of inside and outside IPv4 and IPv6 addresses. Site-to-site tunnels are built using the Internet Protocol Security (IPsec) protocol suite and IKEv1 or IKEv2. After the VPN connection is established, the hosts behind the local gateway can connect to the hosts behind the remote gateway through the secure VPN tunnel.

VPN Basics

Tunneling makes it possible to use a public TCP/IP network, such as the Internet, to create secure connections between remote users and private corporate networks. Each secure connection is called a tunnel.

IPsec-based VPN technologies use the Internet Security Association and Key Management Protocol (ISAKMP, or IKE) and IPsec tunneling standards to build and manage tunnels. ISAKMP and IPsec accomplish the following:

- Negotiate tunnel parameters.
- Establish tunnels.
- Authenticate users and data.
- Manage security keys.
- Encrypt and decrypt data.
- Manage data transfer across the tunnel.
- Manage data transfer inbound and outbound as a tunnel endpoint or router.

A device in a VPN functions as a bidirectional tunnel endpoint. It can receive plain packets from the private network, encapsulate them, create a tunnel, and send them to the other end of the tunnel where they are unencapsulated and sent to their final destination. It can also receive encapsulated packets from the public network, unencapsulate them, and send them to their final destination on the private network.

After the site-to-site VPN connection is established, the hosts behind the local gateway can connect to the hosts behind the remote gateway through the secure VPN tunnel. A connection consists of the IP addresses and hostnames of the two gateways, the subnets behind them, and the method the two gateways use to authenticate to each other.

Hubs—Devices that enable secure VPN connectivity to and from one or more remote branch devices or spokes. Hubs also act as a gateway for spokes to communicate with each other.

Spokes—Devices that connect over VPN to a hub to securely access the corporate resources behind the hub. Spokes communicate with each other through the hub.

Internet Key Exchange (IKE)

Internet Key Exchange (IKE) is a key management protocol that is used to authenticate IPsec peers, negotiate and distribute IPsec encryption keys, and to automatically establish IPsec security associations (SAs).

The IKE negotiation comprises two phases. Phase 1 negotiates a security association between two IKE peers, which enables the peers to communicate securely in Phase 2. During Phase 2 negotiation, IKE establishes SAs for other applications, such as IPsec. Both phases use proposals when they negotiate a connection.

An IKE policy is a set of algorithms that two peers use to secure the IKE negotiation between them. IKE negotiation begins by each peer agreeing on a common (shared) IKE policy. This policy states which security parameters protect subsequent IKE negotiations. For IKE version 1 (IKEv1), IKE policies contain a single set of algorithms and a modulus group. Unlike IKEv1, in an IKEv2 policy, you can select multiple algorithms and modulus groups from which peers can choose during the Phase 1 negotiation. It is possible to create a single IKE policy, although you might want different policies to give higher priority to your most desired options. For site-to-site VPNs, you can create an IKE policy. IKEv1 and IKEv2 each support a maximum of 20 IKE policies, each with a different set of values. Assign a unique priority to each policy that you create. The lower the priority number, the higher the priority.

To define an IKE policy, specify:

- A unique priority (1 to 65,543, with 1 the highest priority).
- An encryption method for the IKE negotiation, to protect the data and ensure privacy.
- A Hashed Message Authentication Codes (HMAC) method (called integrity algorithm in IKEv2) to ensure the identity of the sender, and to ensure that the message has not been modified in transit.
- For IKEv2, a separate pseudorandom function (PRF) used as the algorithm to derive keying material and hashing operations required for the IKEv2 tunnel encryption. The options are the same as those used for the hash algorithm.
- A Diffie-Hellman group to determine the strength of the encryption-key-determination algorithm. The device uses this algorithm to derive the encryption and hash keys.
- An authentication method, to ensure the identity of the peers.
- A limit to the time the device uses an encryption key before replacing it.

When IKE negotiation begins, the peer that starts the negotiation sends all of its policies to the remote peer, and the remote peer searches for a match with its own policies, in priority order. A match between IKE policies exists if they have the same encryption, hash (integrity and PRF for IKEv2), authentication, and Diffie-Hellman values, and an SA lifetime less than or equal to the lifetime in the policy sent. If the lifetimes are not identical, the shorter lifetime—From the remote peer policy—Applies. By default, the Secure Firewall Management Center deploys an IKEv1 policy at the lowest priority for all VPN endpoints to ensure a successful negotiation.

IPsec

IPsec is one of the most secure methods for setting up a VPN. IPsec provides data encryption at the IP packet level, offering a robust security solution that is standards-based. With IPsec, data is transmitted over a public network through tunnels. A tunnel is a secure, logical communication path between two peers. Traffic that enters an IPsec tunnel is secured by a combination of security protocols and algorithms.

An IPsec Proposal policy defines the settings required for IPsec tunnels. An IPsec proposal is a collection of one or more crypto-maps that are applied to the VPN interfaces on the devices. A crypto map combines all the components required to set up IPsec security associations, including:

- A proposal (or transform set) is a combination of security protocols and algorithms that secure traffic in an IPsec tunnel. During the IPsec security association (SA) negotiation, peers search for a proposal that is the same at both peers. When it is found, it is applied to create an SA that protects data flows in the access list for that crypto map, protecting the traffic in the VPN. There are separate IPsec proposals for IKEv1 and IKEv2. In IKEv1 proposals (or transform sets), for each parameter, you set one value. For IKEv2 proposals, you can configure multiple encryption and integration algorithms for a single proposal.
- A crypto map, combines all components required to set up IPsec security associations (SA), including IPsec rules, proposals, remote peers, and other parameters that are necessary to define an IPsec SA. When two peers try to establish an SA, they must each have at least one compatible crypto map entry.

  Dynamic crypto map policies are used in site-to-site VPNs when an unknown remote peer tries to start an IPsec security association with the local hub. The hub cannot be the initiator of the security association negotiation. Dynamic crypto-policies allow remote peers to exchange IPsec traffic with a local hub even if the hub does not know the remote peer's identity. A dynamic crypto map policy essentially creates a crypto map entry without all the parameters configured. The missing parameters are later dynamically configured (as the result of an IPsec negotiation) to match a remote peer's requirements.

  Dynamic crypto map policies are applicable to both hub-and-spoke and point-to-point VPN topologies. To apply dynamic crypto map policies, specify a dynamic IP address for one of the peers in the topology and ensure that the dynamic crypto-map is enabled on this topology. Note that in a full mesh VPN topology, you can apply only static crypto map policies.

Note Simultaneous IKEv2 dynamic crypto map is not supported for the same interface for both remote access and site-to-site VPNs on Firewall Threat Defense.

VPN Packet Flow

On a Firewall Threat Defense device, by default no traffic is allowed to pass through access-control without explicit permission. VPN tunnel traffic as well, is not relayed to the endpoints until it has passed through Snort. Incoming tunnel packets are decrypted before being sent to the Snort process. Snort processes outgoing packets before encryption.

Access Control identifying the protected networks for each endpoint node of a VPN tunnel determines which traffic is allowed to pass through the Firewall Threat Defense device and reach the endpoints. For Remote Access VPN traffic, a Group Policy filter or an Access Control rule must be configured to permit VPN traffic flow.

In addition, the system does not send tunnel traffic to the public source when the tunnel is down.

IPsec Flow Offload

You can configure supporting device models to use IPsec flow offload. After the initial setup of an IPsec site-to-site VPN or remote access VPN security association (SA), IPsec connections are offloaded to the

field-programmable gate array (FPGA) in the device, which should improve device performance. On the Secure Firewall 1200 series, IPsec connections are offloaded to the Marvell Cryptographic Accelerator (CPT) to improve device performance.On the Secure Firewall 6100 series, IPsec connections are offloaded to the Kintex 7 (KC400) FPGA. This FPGA contains a built-in crypto engine that is capable of handling AES-GCM-128 and AES-GCM-256 encryption and decryption.

Offloaded operations specifically relate to the pre-decryption and decryption processing on ingress, and the pre-encryption and encryption processing on egress. The system software handles the inner flow to apply your security policies.

IPsec flow offload is enabled by default, and applies to the following device types:

- Secure Firewall 1200
- Secure Firewall 3100
- Secure Firewall 4200
- Secure Firewall 6100

IPsec flow offload is also used when the device's VTI loopback interface is enabled.

For asymmetric flows in cluster distributed site-to-site VPN mode, IPsec flow offload now lets the flow owner decrypt IPsec traffic in hardware that was forwarded over the cluster control link. This feature is not configurable and is always available with IPsec flow offload.

Limitations for IPsec Flow Offload

The following IPsec flows are not offloaded:

- IKEv1 tunnels. Only IKEv2 tunnels will be offloaded. IKEv2 supports stronger ciphers.
- Flows that have volume-based rekeying configured.
- Flows that have compression configured.
- Transport mode flows. Only tunnel mode flows will be offloaded.
- AH format. Only ESP/NAT-T format will be supported.
- Flows that have post-fragmentation configured.
- Flows that have anti-replay window size other than 64bit and anti-replay is not disabled.
- Flows that have firewall filter enabled.
- Mult-instance mode.
- Secure Firewall 6100 supports AES-GCM-128 and AES-GCM-256 ciphers only. IPsec tunnels that are configured with other ciphers are not offloaded. Any IPsec packets that are not offloaded are processed by the software engine in the CPU.

Configure IPsec Flow Offload

IPsec flow offload is enabled by default on hardware platforms that support the feature. To change the configuration, use FlexConfig to implement the **flow-offload-ipsec** command. See the ASA command reference for detailed information about the command.

VPN Licensing

There is no specific licensing for enabling Secure Firewall Threat Defense VPN, it is available by default.

The Firewall Management Center determines whether to allow or block the usage of strong crypto on the Firewall Threat Defense device based on attributes provided by the smart licensing server.

This is controlled by whether you selected the option to allow export-controlled functionality on the device when you registered with the Cisco Smart License Manager. If you are using the evaluation license, or you did not enable export-controlled functionality, you cannot use strong encryption.

If you have created your VPN configurations with an evaluation license, and upgrade your license from evaluation to smart license with export-controlled functionality, check, and update your encryption algorithms for stronger encryption and for the VPNs to work properly. DES-based encryptions are no longer supported.

How Secure Should a VPN Connection Be?

Because a VPN tunnel typically traverses a public network, most likely the Internet, you need to encrypt the connection to protect the traffic. You define the encryption and other security techniques to apply using IKE polices and IPsec proposals.

If your device license allows you to apply strong encryption, there is a wide range of encryption and hash algorithms, and Diffie-Hellman groups, from which to choose. However, as a general rule, the stronger the encryption that you apply to the tunnel, the worse the system performance. Find a balance between security and performance that provides sufficient protection without compromising efficiency.

We cannot provide specific guidance on which options to choose. If you operate within a larger corporation or other organization, there might already be defined standards that you need to meet. If not, take the time to research the options.

The following topics explain the available options.

Complying with Security Certification Requirements

Many VPN settings have options that allow you to comply with various security certification standards. Review your certification requirements and the available options to plan your VPN configuration.

Deciding Which Encryption Algorithm to Use

When deciding which encryption algorithms to use for the IKE policy or IPsec proposal, your choice is limited to algorithms supported by the devices in the VPN.

For IKEv2, you can configure multiple encryption algorithms. The system orders the settings from the most secure to the least secure and negotiates with the peer using that order. For IKEv1, you can select a single option only.

For IPsec proposals, the algorithm is used by the Encapsulating Security Protocol (ESP), which provides authentication, encryption, and anti-replay services. ESP is IP protocol type 50. In IKEv1 IPsec proposals, the algorithm name is prefixed with ESP-.

If your device license qualifies for strong encryption, you can choose from the following encryption algorithms. If you are not qualified for strong encryption, you can select DES only.

Note If you are qualified for strong encryption, before upgrading from the evaluation license to a smart license, check and update your encryption algorithms for stronger encryption so that the VPN configuration works properly. Choose AES-based algorithms. DES is not supported if you are registered using an account that supports strong encryption. After registration, you cannot deploy changes until you remove all uses of DES.

- AES-GCM—(IKEv2 only.) Advanced Encryption Standard in Galois/Counter Mode is a block cipher mode of operation providing confidentiality and data-origin authentication, and provides greater security than AES. AES-GCM offers three different key strengths: 128-, 192-, and 256-bit keys. A longer key provides higher security but a reduction in performance. GCM is a mode of AES that is required to support NSA Suite B. NSA Suite B is a set of cryptographic algorithms that devices must support to meet federal standards for cryptographic strength. .
- AES—Advanced Encryption Standard is a symmetric cipher algorithm that provides greater security than DES and is computationally more efficient than 3DES. AES offers three different key strengths: 128-, 192-, and 256-bit keys. A longer key provides higher security but a reduction in performance.
- DES—Data Encryption Standard, which encrypts using 56-bit keys, is a symmetric secret-key block algorithm. If your license account does not meet the requirements for export controls, this is your only option.
- Null, ESP-Null—A null encryption algorithm provides authentication without encryption. This method is not secure; use at your own discretion.

Deciding Which Hash Algorithms to Use

In IKE policies, the hash algorithm creates a message digest, which is used to ensure message integrity. In IKEv2, the hash algorithm is separated into two options, one for the integrity algorithm, and one for the pseudo-random function (PRF).

In IPsec proposals, the hash algorithm is used by the Encapsulating Security Protocol (ESP) for authentication. In IKEv2 IPsec Proposals, this is called the integrity hash. In IKEv1 IPsec proposals, the algorithm name is prefixed with ESP-, and there is also an -HMAC suffix (which stands for “hash method authentication code”).

For IKEv2, you can configure multiple hash algorithms. The system orders the settings from the most secure to the least secure and negotiates with the peer using that order. For IKEv1, you can select a single option only.

You can choose from the following hash algorithms.

- SHA (Secure Hash Algorithm)—Standard SHA (SHA1) produces a 160-bit digest.

  The following SHA-2 options, which are even more secure, are available for IKEv2 configurations. Choose one of these if you want to implement the NSA Suite B cryptography specification.

  - SHA256—Specifies the Secure Hash Algorithm SHA 2 with the 256-bit digest.
  - SHA384—Specifies the Secure Hash Algorithm SHA 2 with the 384-bit digest.
  - SHA512—Specifies the Secure Hash Algorithm SHA 2 with the 512-bit digest.

- Null or None (NULL, ESP-NONE)—(IPsec Proposals only.) A null Hash Algorithm; this is typically used for testing purposes only. However, you should choose the null integrity algorithm if you select one of the AES-GCM options as the encryption algorithm. Even if you choose a non-null option, the integrity hash is ignored for these encryption standards.

Overview of Additional Key Exchanges

A key exchange in IKEv2 computes a shared secret during the setup of a Security Association (SA). Additional key exchanges in IKEv2 secure IPsec communication from quantum computer attacks. IKEv2 can use Diffie-Hellman (DH) groups, or Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) groups for the additional key exchanges. Each exchange uses different DH or ML-KEM groups. The computed shared secret for the SA setup is a combination of all the keys derived from each exchange. An IKE SA is established after the multiple key exchanges between the IKE peers.

You can configure a maximum of seven multiple key exchanges in an IKEv2 policy and IKEv2 IPsec proposal. Firewall Threat Defense encrypts the intermediate key exchanges using the keys derived from the previous exchange. If the initiator and responder peers do not agree on an algorithm, the negotiation fails and an error notification is sent to the initiator. You can also configure the transform as none. If you choose none, the key exchange does not happen.

After you configure additional key exchanges in an IKEv2 policy and IKEv2 IPsec proposal, you must add the policy or proposal to the required site-to-site VPN or SD-WAN VPN topology. Note that you can use these IKEv2 policies and IKEv2 IPsec proposals for an RA VPN policy.

Firewall Management Center supports these algorithms for additional key exchanges:

- 14—Diffie-Hellman group 14 (2048-bit)
- 15—Diffie-Hellman group 15 (3072-bit)
- 16—Diffie-Hellman group 16 (4096-bit)
- 19—Diffie-Hellman group 19 (256-bit)
- 20—Diffie-Hellman group 20 (384-bit)
- 21—Diffie-Hellman group 21 (521-bit)
- 31—Diffie-Hellman group 31 (256-bit)
- 35—Module-Lattice group 35 (ML-KEM-512: 512-bit)
- 36—Module-Lattice group 36 (ML-KEM-768: 768-bit)
- 37—Module-Lattice group 37 (ML-KEM-1024: 1024-Bit)

For more information about configuring additional key exchanges in an IKEv2 policy or IKEv2 IPsec proposal, see Configure IKEv2 Policy Objects and Configure IKEv2 IPsec Proposal Objects.

Prerequisites

Supported Firewall Threat Defense versions for additional key exchanges are:

- Version 10.0 and later for enabling DH and ML-KEM algorithms.
- Version 7.4.1 and later for only enabling DH algorithms.

Deciding Which Diffie-Hellman Modulus Group to Use

You can use the following Diffie-Hellman key derivation algorithms to generate IPsec security association (SA) keys. Each group has a different size modulus. A larger modulus provides higher security, but requires more processing time. You must have a matching modulus group on both peers.

If you select AES encryption, to support the large key sizes required by AES, you should use Diffie-Hellman (DH) Group 5 or higher. IKEv1 policies do not support all of the groups listed below.

To implement the NSA Suite B cryptography specification, use IKEv2 and select one of the elliptic curve Diffie-Hellman (ECDH) options: 19, 20, or 21. Elliptic curve options and groups that use 2048-bit modulus are less exposed to attacks such as Logjam.

For IKEv2, you can configure multiple groups. The system orders the settings from the most secure to the least secure and negotiates with the peer using that order. For IKEv1, you can select a single option only.

- 14—Diffie-Hellman Group 14: 2048-bit modular exponential (MODP) group. Considered good protection for 192-bit keys.
- 15—Diffie-Hellman Group 15: 3072-bit MODP group.
- 16—Diffie-Hellman Group 16: 4096-bit MODP group.
- 19—Diffie-Hellman Group 19: National Institute of Standards and Technology (NIST) 256-bit elliptic curve modulo a prime (ECP) group.
- 20—Diffie-Hellman Group 20: NIST 384-bit ECP group.
- 21—Diffie-Hellman Group 21: NIST 521-bit ECP group.
- 31—Diffie-Hellman Group 31: Curve25519 256-bit EC Group.

Deciding Which Authentication Method to Use

Preshared keys and digital certificates are the methods of authentication available for VPNs.

Site-to-site, IKEv1 and IKEv2 VPN connections can use both options.

Remote Access, which uses SSL and IPsec IKEv2 only, supports digital certificate authentication only.

Preshared keys allow for a secret key to be shared between two peers and used by IKE during the authentication phase. The same shared key must be configured at each peer or the IKE SA cannot be established.

Digital certificates use RSA key pairs to sign and encrypt IKE key management messages. Certificates provide non-repudiation of communication between two peers, meaning that it can be proved that the communication actually took place. When using this authentication method, you need a Public Key Infrastructure (PKI) defined where peers can obtain digital certificates from a Certification Authority (CA). CAs manage certificate requests and issue certificates to participating network devices providing centralized key management for all of the participating devices.

Preshared keys do not scale well, using a CA improves the manageability and scalability of your IPsec network. With a CA, you do not need to configure keys between all encrypting devices. Instead, each participating device is registered with the CA, and requests a certificate from the CA. Each device that has its own certificate and the public key of the CA can authenticate every other device within a given CA's domain.

Pre-shared Keys

Pre-shared key enables you to share a secret key between two peers. IKE uses the key in the authentication phase. You must configure the same shared key on each peer, or the IKE SA cannot be established.

To configure the pre-shared keys, choose whether you want to use a manual or automatically generated key, and then specify the key in the IKEv1/IKEv2 options. Then, when you deploy your configuration, the key is configured on all devices in the topology.

PKI Infrastructure and Digital Certificates

Public Key Infrastructure

A PKI provides centralized key management for participating network devices. It is a defined set of policies, procedures, and roles that support *public key cryptography* by generating, verifying, and revoking *public key certificates* commonly known as *digital certificates*.

In public key cryptography, each endpoint of a connection has a key pair consisting of both a public and a private key. The key pairs are used by the VPN endpoints to sign and encrypt messages. The keys act as complements, and anything encrypted with one of the keys can be decrypted with the other, securing the data flowing over the connection.

Generate a general purpose RSA, ECDSA, or EDDSA key pair, used for both signing and encryption, or you generate separate key pairs for each purpose. Separate signing and encryption keys help to reduce exposure of the keys. SSL uses a key for encryption but not signing, however, IKE uses a key for signing but not encryption. By using separate keys for each, exposure of the keys is minimized.

Digital Certificates or Identify Certificates

When you use Digital Certificates as the authentication method for VPN connections, peers are configured to obtain digital certificates from a Certificate Authority (CA). CAs are trusted authorities that "sign" certificates to verify their authenticity, thereby guaranteeing the identity of the device or user.

CA servers manage public CA certificate requests and issue certificates to participating network devices as part of a Public Key Infrastructure (PKI), this activity is called Certificate Enrollment. These digital certificates, also called identity certificates contain:

- The digital identification of the owner for authentication, such as name, serial number, company, department, or IP address.
- A public key needed to send and receive encrypted data to the certificate owner.
- The secure digital signature of a CA.

Certificates also provide non-repudiation of communication between two peers, meaning that it they prove that the communication actually took place.

Certificate Enrollment

Using a PKI improves the manageability and scalability of your VPN since you do not have to configure pre-shared keys between all the encrypting devices. Instead, you individually *enroll* each participating device with a CA server, which is explicitly trusted to validate identities and create an identity certificate for the device. When this has been accomplished, each participating peer sends their identity certificate to the other peer to validate their identities and establish encrypted sessions with the public keys contained in the certificates. See Certificate Enrollment Objects for details on enrolling Firewall Threat Defense devices.

Certificate Authority Certificates

In order to validate a peer's certificate, each participating device must retrieve the CA's certificate from the server. A CA certificate is used to sign other certificates. It is self-signed and called a root certificate. This certificate contains the public key of the CA, used to decrypt and validate the CA's digital signature and the contents of the received peer's certificate. The CA certificate may be obtained by:

- Using the Simple Certificate Enrollment Protocol (SCEP) or Enrollment over Secure Transport (EST) to retrieve the CA's certificate from the CA server
- Manually copying the CA's certificate from another participating device

Trustpoints

Once enrollment is complete, a trustpoint is created on the managed device. It is the object representation of a CA and associated certificates. A trustpoint includes the identity of the CA, CA-specific parameters, and an association with a single enrolled identity certificate.

PKCS#12 File

A PKCS#12, or PFX, file holds the server certificate, any intermediate certificates, and the private key in one encrypted file. This type of file may be imported directly into a device to create a trustpoint.

Revocation Checking

A CA may also revoke certificates for peers that no longer participate in you network. Revoked certificates are either managed by an Online Certificate Status Protocol (OCSP) server or are listed in a certificate revocation list (CRL) stored on an LDAP server. A peer may check these before accepting a certificate from another peer.

Removed or Deprecated Hash Algorithms, Encryption Algorithms, and Diffie-Hellman Modulus Groups

Support has been removed for less secure ciphers. We recommend that you update your VPN configuration before you upgrade to Firewall Threat Defense 6.70 to supported DH and encryption algorithms to ensure the VPN works correctly.

Update your IKE proposals and IPSec policies to match the ones supported in Firewall Threat Defense 6.70 and then deploy the configuration changes.

The following less secure ciphers have been removed or deprecated in Firewall Threat Defense 6.70 onwards:

- **Diffie-Hellman GROUP 5** is deprecated for IKEv1 and IKEv2.
- Diffie-Hellman groups 2 and 24 have been removed.
- **Encryption algorithms**: 3DES, AES-GMAC, AES-GMAC-192, AES-GMAC-256 have been removed.

Note **DES** continues to be supported in evaluation mode or for users who do not satisfy export controls for strong encryption.

NULL is removed in IKEv2 policy, but supported in both IKEv1 and IKEv2 IPsec transform-sets.

VPN Topology Options

When you create a new VPN topology you must, at minimum, give it a unique name, specify a topology type, and select the IKE version. You can select from three types of topologies, each containing a group of VPN tunnels:

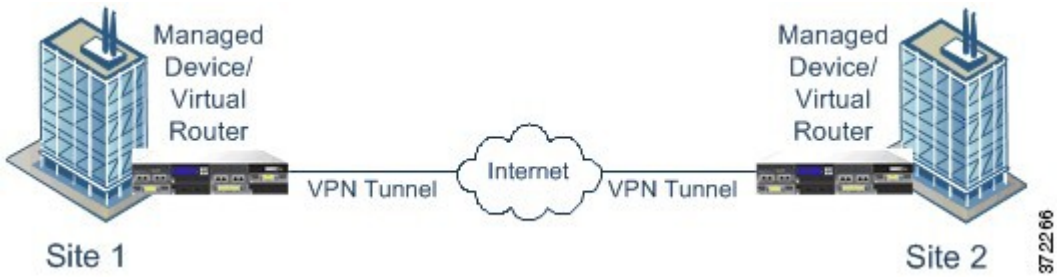
- Point-to-point (PTP) topologies establish a VPN tunnel between two endpoints.
- Hub and Spoke topologies establish a group of VPN tunnels connecting a hub endpoint to a group of spoke endpoints.
- Full Mesh topologies establish a group of VPN tunnels among a set of endpoints.

Define a pre-shared key for VPN authentication manually or automatically, there is no default key. When choosing automatic, the Secure Firewall Management Center generates a pre-shared key and assigns it to all the nodes in the topology.

Point-to-Point VPN Topology

In a point-to-point VPN topology, two endpoints communicate directly with each other. You configure the two endpoints as peer devices, and either device can start the secured connection.

The following diagram displays a typical point-to-point VPN topology.



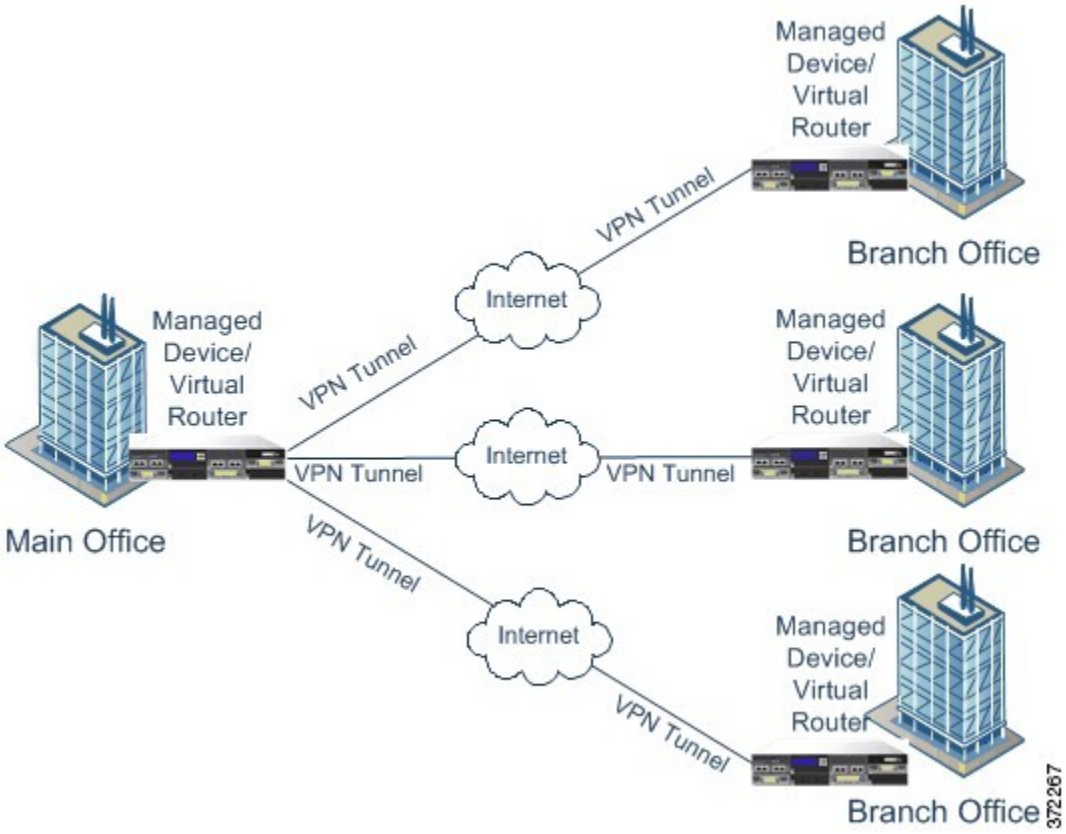
Hub and Spoke VPN Topology

In a Hub and Spoke VPN topology, a central endpoint (hub node) connects with multiple remote endpoints (spoke nodes). Each connection between the hub node and an individual spoke endpoint is a separate VPN tunnel. The hosts behind any of the spoke nodes can communicate with each other through the hub node.

The Hub and Spoke topology commonly represent a VPN that connects an organization's main and branch office locations using secure connections over the Internet or other third-party network. These deployments

provide all employees with controlled access to the organization's network. Typically, the hub node is located at the main office. Spoke nodes are located at branch offices and start most of the traffic.

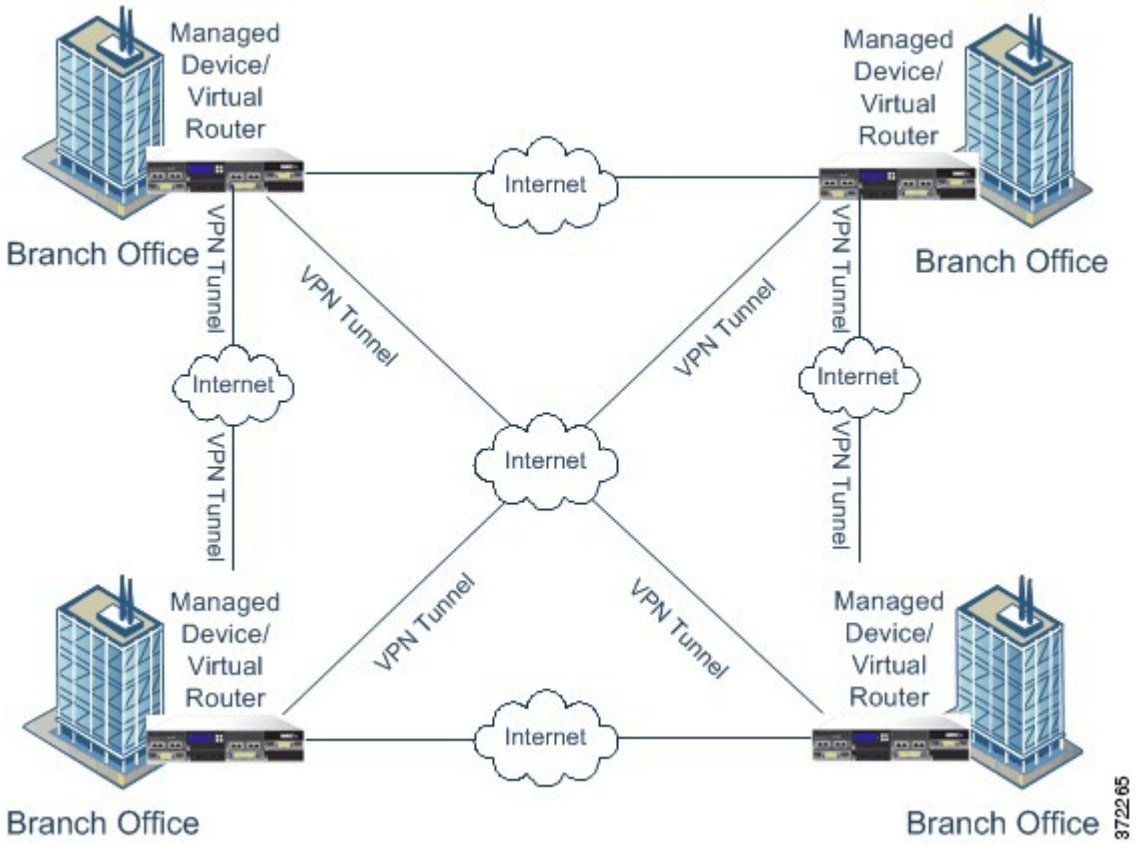
The following diagram displays a typical Hub and Spoke VPN topology.



Full Mesh VPN Topology

In a Full Mesh VPN topology, all endpoints can communicate with every other endpoint by an individual VPN tunnel. This topology offers redundancy so that when one endpoint fails, the remaining endpoints can still communicate with each other. It commonly represents a VPN that connects a group of decentralized branch office locations. The number of VPN-enabled managed devices you deploy in this configuration depends on the level of redundancy you require.

The following diagram displays a typical Full Mesh VPN topology.



Implicit Topologies

In addition to the three main VPN topologies, other more complex topologies can be created as combinations of these topologies. They include:

- Partial mesh—A network in which some devices are organized in a full mesh topology, and other devices form either a hub-and-spoke or a point-to-point connection to some of the fully meshed devices. A partial mesh does not provide the level of redundancy of a full mesh topology, but it is less expensive to implement. Partial mesh topologies are used in peripheral networks that connect to a fully meshed backbone.
- Tiered hub-and-spoke—A network of hub-and-spoke topologies in which a device can behave as a hub in one or more topologies and a spoke in other topologies. Traffic is permitted from spoke groups to their most immediate hub.
- Joined hub-and-spoke—A combination of two topologies (hub-and-spoke, point-to-point, or full mesh) that connect to form a point-to-point tunnel. For example, a joined hub-and-spoke topology could comprise two hub-and-spoke topologies, with the hubs acting as peer devices in a point-to-point topology.

VPN Troubleshooting

This chapter describes the different methods to troubleshoot VPN tunnels in Firewall Threat Defense.

System Messages

The Message Center is the place to start your troubleshooting. This feature allows you to view messages that are continually generated about system activities and status. To open the Message Center, click **System Status**, located to the immediate right of the **Deploy** button in the main menu.

VPN System Logs

You can enable logging of VPN troubleshoot syslogs for Firewall Threat Defense devices. Logging information can help you identify and isolate network or device configuration problems. When you enable VPN logging, the Firewall Threat Defense devices send VPN syslogs to the Firewall Management Center.

All VPN syslogs appear with a default severity level **errors** or a higher severity (unless changed). You can manage the VPN logging through Firewall Threat Defense platform settings. You can adjust the message severity level by editing the **VPN Logging Settings** in the Firewall Threat Defense platform settings policy for targeted devices. See Configure Syslog Logging for Firewall Threat Defense Devices for details on enabling VPN logging, configuring syslog servers, and viewing the system logs.

From the Troubleshooting Logs table (**Troubleshooting** > **+ Show more** > **Advanced** > **Troubleshooting Logs**), you can view and analyze the VPN syslog messages to identify and isolate issues with your network and device configuration.

We recommend that you set the logging level of the VPN logs as level 3 (Errors). Setting the VPN logging level to level 4 and above (Warnings, Notifications, Informational or Debugging) could overload the Firewall Management Center.

Note When you configure a device with site-to-site or remote access VPN, it automatically enables sending VPN syslogs to the Firewall Management Center by default.

Debug Commands

This section explains how you use debug commands to help you diagnose and resolve VPN-related problems. The commands described here are not exhaustive, this section include commands according to their usefulness in assisting you to diagnose VPN-related problems.

Usage Guidelines

Debug commands consume high-priority CPU resources, which can make the system unusable. Only use debug commands for specific troubleshooting or when instructed by Cisco TAC. To minimize impact, run these commands during periods of low network traffic.

You can view debug output in a CLI session only. Output is directly available when connected to the Console port, or when in the diagnostic CLI (enter **system support diagnostic-cli**). You can also view output from the device CLI with **show console-output** command.

To show debugging messages for a given feature, use the **debug** command. To disable the display of debug messages, use the **no** form of this command. Use **no debug all** to turn off all debugging commands.

debug *feature* [*subfeature*] [*level*]
no debug *feature* [*subfeature*]

Syntax Description

feature	Specifies the feature for which you want to enable debugging. To see the available features, use the **debug ?** command for CLI help.
subfeature	(Optional) Depending on the feature, you can enable debug messages for one or more subfeatures. Use ? to see the available subfeatures.
level	(Optional) Specifies the debugging level. Use ? to see the available levels.

Command Default

The default debugging level is 1.

Example

With multiple sessions running on remote access VPN, troubleshooting can be difficult, given the size of the logs. You can use the **debug webvpn condition** command to set up filters to target your debug process more precisely.

debug webvpn condition {**group** *name* | **p-ipaddress** *ip_address* [{**subnet** *subnet_mask* | **prefix** *length*}] | **reset** | **user** *name*}

Where:

- **group** *name* filters on a group policy (not a tunnel group or connection profile).
- **p-ipaddress** *ip_address* [{**subnet** *subnet_mask* | **prefix** *length*}] filters on the public IP address of the client. The subnet mask (for IPv4) or prefix (for IPv6) is optional.
- **reset** resets all filters. You can use the **no debug webvpn condition** command to turn off a specific filter.
- **user** *name* filters by username.

If you configure more than one condition, the conditions are conjoined (ANDed), so that debugs appear only if all conditions are met.

After setting up the condition filter, use the base **debug webvpn** command to turn on the debug. Setting the conditions alone does not enable the debug. Use the **show debug** and **show webvpn debug-condition** commands to view the current state of debugging.

The following shows an example of enabling a conditional debug on the user jdoe.

```
firepower# debug webvpn condition user jdoe

firepower# show webvpn debug-condition
INFO: Webvpn conditional debug is turned ON
INFO: User name filters:
INFO: jdoe

firepower# debug webvpn
INFO: debug webvpn  enabled at level 1.

firepower# show debug
debug webvpn  enabled at level 1
INFO: Webvpn conditional debug is turned ON
INFO: User name filters:
INFO: jdoe
```

Related Commands

Command	Description
show debug	Shows the currently active debug settings.
undebug	Disables debugging for a feature. This command is a synonym for **no debug**.

debug aaa

See the following commands for debugging configurations or authentication, authorization, and accounting (AAA) settings.

debug *aaa* [*accounting* | *authentication* | *authorization* | *common* | *internal* | *shim* | *url-redirect*]

Syntax Description

aaa	Enables debugging for AAA. Use ? to see the available subfeatures.
accounting	(Optional) Enables AAA accounting debugging.
authentication	(Optional) Enables AAA authentication debugging.
authorization	(Optional) Enables AAA authorization debugging.
common	(Optional) Specifies the AAA common debug level. Use ? to see the available levels.
internal	(Optional) Enables AAA internal debugging.
shim	(Optional) Specifies the AAA shim debug level. Use ? to see the available levels.
url-redirect	(Optional) Enables AAA url-redirect debugging.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug aaa	Shows the currently active debug settings for AAA.
undebug aaa	Disables debugging for AAA. This command is a synonym for **no debug aaa**.

debug crypto

See the following commands for debugging configurations or settings associated with crypto.

debug *crypto* [*ca* | *condition* | *engine* | *ike-common* | *ikev1* | *ikev2* | *ipsec* | *ss-apic*]

Syntax Description

crypto	Enables debugging for *crypto*. Use ? to see the available subfeatures.
ca	(Optional) Specifies the PKI debug levels. Use ? to see the available subfeatures.
condition	(Optional) Specifies the IPsec/ISAKMP debug filters. Use ? to see the available filters.

engine	(Optional) Specifies the crypto engine debug levels. Use ? to see the available levels.
ike-common	(Optional) Specifies the IKE common debug levels. Use ? to see the available levels.
ikev1	(Optional) Specifies the IKE version 1 debug levels. Use ? to see the available levels.
ikev2	(Optional) Specifies the IKE version 2 debug levels. Use ? to see the available levels.
ipsec	(Optional) Specifies the IPsec debug levels. Use ? to see the available levels.
condition	(Optional) Specifies the Crypto Secure Socket API debug levels. Use ? to see the available levels.
vpnclient	(Optional) Specifies the EasyVPN client debug levels. Use ? to see the available levels.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug crypto	Shows the currently active debug settings for crypto.
undebug crypto	Disables debugging for crypto. This command is a synonym for **no debug crypto**.

debug crypto ca

See the following commands for debugging configurations or settings associated with crypto ca.

debug *crypto ca* [*cluster* | *messages* | *periodic-authentication* | *scep-proxy* | *transactions* | *trustpool*] [*1-255*]

Syntax Description

crypto ca	Enables debugging for *crypto ca*. Use ? to see the available subfeatures.
cluster	(Optional) Specifies the PKI cluster debug level. Use ? to see the available levels.
cmp	(Optional) Specifies the CMP transactions debug level. Use ? to see the available levels.
messages	(Optional) Specifies the PKI Input/Output message debug level. Use ? to see the available levels.
periodic-authentication	(Optional) Specifies the PKI periodic-authentication debug level. Use ? to see the available levels.
scep-proxy	(Optional) Specifies the SCEP proxy debug level. Use ? to see the available levels.
server	(Optional) Specifies the local CA server debug level. Use ? to see the available levels.

transactions	(Optional) Specifies the PKI transaction debug level. Use ? to see the available levels.
trustpool	(Optional) Specifies the trustpool debug level. Use ? to see the available levels.
1-255	(Optional) Specifies the debugging level.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug crypto ca	Shows the currently active debug settings for crypto ca.
undebug	Disables debugging for crypto ca. This command is a synonym for **no debug crypto ca**.

debug crypto ikev1

See the following commands for debugging configurations or settings associated with Internet Key Exchange version 1 (IKEv1).

debug *crypto ikev1* [*timers*] [*1-255*]

Syntax Description

ikev1	Enables debugging for *ikev1*. Use ? to see the available subfeatures.
timers	(Optional) Enables debugging for IKEv1 timers.
1-255	(Optional) Specifies the debugging level.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug crypto ikev1	Shows the currently active debug settings for IKEv1.
undebug crypto ikev1	Disables debugging for IKEv1. This command is a synonym for **no debug crypto ikev1**.

debug crypto ikev2

See the following commands for debugging configurations or settings associated with Internet Key Exchange version 2 (IKEv2).

debug *crypto ikev2* [*ha* | *platform* | *protocol* | *timers*]

Syntax Description

ikev2	Enables debugging *ikev2*. Use ? to see the available subfeatures.
ha	(Optional) Specifies the IKEv2 HA debug level. Use ? to see the available levels.
platform	(Optional) Specifies the IKEv2 platform debug level. Use ? to see the available levels.

protocol	(Optional) Specifies the IKEv2 protocol debug level. Use ? to see the available levels.
timers	(Optional) Enables debugging for IKEv2 timers.

Command Default The default debugging level is 1.

Related Commands

Command	Description
show debug crypto ikev2	Shows the currently active debug settings for IKEv2.
undebugcrypto ikev2	Disables debugging for IKEv2. This command is a synonym for **no debug crypto ikev2**.

debug crypto ipsec

See the following commands for debugging configurations or settings associated with IPsec.

debug *crypto ipsec* [*1-255*]

Syntax Description

ipsec	Enables debugging for *ipsec*. Use ? to see the available subfeatures.
1-255	(Optional) Specifies the debugging level.

Command Default The default debugging level is 1.

Related Commands

Command	Description
show debug crypto ipsec	Shows the currently active debug settings for IPsec.
undebugcrypto ipsec	Disables debugging for IPsec. This command is a synonym for **no debug crypto ipsec**.

debug ldap

See the following commands for debugging configurations or settings associated with LDAP (Lightweight Directory Access Protocol).

debug *ldap* [*1-255*]

Syntax Description

ldap	Enables debugging for LDAP. Use ? to see the available subfeatures.
1-255	(Optional) Specifies the debugging level.

Command Default The default debugging level is 1.

Related Commands

Command	Description
show debug ldap	Shows the currently active debug settings for LDAP.

Command	Description
undebugldap	Disables debugging for LDAP. This command is a synonym for **no debug ldap**.

debug ssl

See the following commands for debugging configurations or settings associated with SSL sessions.

debug *ssl* [*cipher* | *device*] [*1-255*]

Syntax Description

ssl	Enables debugging for SSL. Use ? to see the available subfeatures.
cipher	(Optional) Specifies the SSL cipher debug level. Use ? to see the available levels.
device	(Optional) Specifies the SSL device debug level. Use ? to see the available levels.
1-255	(Optional) Specifies the debugging level.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug ssl	Shows the currently active debug settings for SSL.
undebug ssl	Disables debugging for SSL. This command is a synonym for **no debug ssl**.

debug webvpn

See the following commands for debugging configurations or settings associated with WebVPN.

debug *webvpn* [*anyconnect* | *chunk* | *cifs* | *citrix* | *compression* | *condition* | *cstp-auth* | *customization* | *failover* | *html* | *javascript* | *kcd* | *listener* | *mus* | *nfs* | *request* | *response* | *saml* | *session* | *task* | *transformation* | *url* | *util* | *xml*]

Syntax Description

webvpn	Enables debugging for WebVPN. Use ? to see the available subfeatures.
anyconnect	(Optional) Specifies the WebVPN Secure Client debug level. Use ? to see the available levels.
chunk	(Optional) Specifies the WebVPN chunk debug level. Use ? to see the available levels.
cifs	(Optional) Specifies the WebVPN CIFS debug level. Use ? to see the available levels.
citrix	(Optional) Specifies the WebVPN Citrix debug level. Use ? to see the available levels.
compression	(Optional) Specifies the WebVPN compression debug level. Use ? to see the available levels.

condition	(Optional) Specifies the WebVPN filter conditions debug level. Use ? to see the available levels.
cstp-auth	(Optional) Specifies the WebVPN CSTP authentication debug level. Use ? to see the available levels.
customization	(Optional) Specifies the WebVPN customization debug level. Use ? to see the available levels.
failover	(Optional) Specifies the WebVPN failover debug level. Use ? to see the available levels.
html	(Optional) Specifies the WebVPN HTML debug level. Use ? to see the available levels.
javascript	(Optional) Specifies the WebVPN Javascript debug level. Use ? to see the available levels.
kcd	(Optional) Specifies the WebVPN KCD debug level. Use ? to see the available levels.
listener	(Optional) Specifies the WebVPN listener debug level. Use ? to see the available levels.
mus	(Optional) Specifies the WebVPN MUS debug level. Use ? to see the available levels.
nfs	(Optional) Specifies the WebVPN NFS debug level. Use ? to see the available levels.
request	(Optional) Specifies the WebVPN request debug level. Use ? to see the available levels.
response	(Optional) Specifies the WebVPN response debug level. Use ? to see the available levels.
saml	(Optional) Specifies the WebVPN SAML debug level. Use ? to see the available levels.
session	(Optional) Specifies the WebVPN session debug level. Use ? to see the available levels.
task	(Optional) Specifies the WebVPN task debug level. Use ? to see the available levels.
transformation	(Optional) Specifies the WebVPN transformation debug level. Use ? to see the available levels.
url	(Optional) Specifies the WebVPN URL debug level. Use ? to see the available levels.
util	(Optional) Specifies the WebVPN utility debug level. Use ? to see the available levels.

xml	(Optional) Specifies the WebVPN XML debug level. Use ? to see the available levels.

Command Default

The default debugging level is 1.

Related Commands

Command	Description
show debug webvpn	Shows the currently active debug settings for WebVPN.
undebug webvpn	Disables debugging for WebVPN. This command is a synonym for **no debug webvpn**.