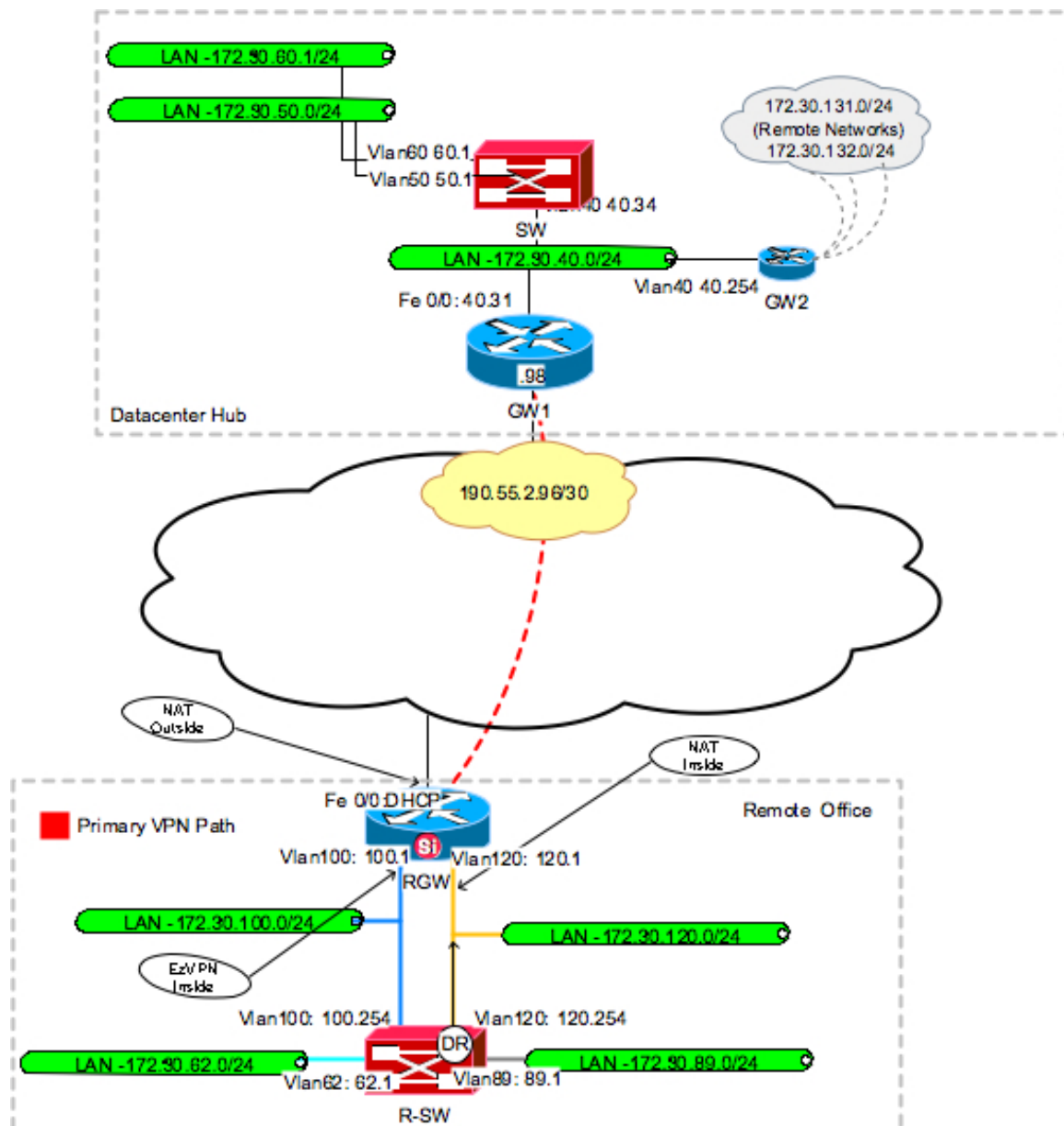


Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

Previously in our series on building router-based VPN gateways, we learned how to support topologies by building a gateway with Cisco EzVPN gateway to support a network-to-network IPsec VPN. We also covered the hardware client configuration for full-crypto peering on the EzVPN gateway. Now we'll move on to split-tunnel client configuration, which is more efficient and secure, but a bit more complex.

The scenario illustrated below could serve as a primary link solution for a small remote office or as a secondary or backup path solution for an office that already has a primary leased line or another VPN link. It uses split tunneling to secure the network connections, rather than the simpler full-crypto solution discussed earlier.



Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

Split tunneling is considered to be more efficient than a full-crypto VPN topology, but that efficiency comes at a price. That's because split tunneling does not behave as expected when used in conjunction with Cisco's EzVPN hardware client running in network extension mode. Normally, when a split-tunnel policy is used with a software or hardware client, traffic that is designated to be secured is encrypted and the "unsecured" traffic is forwarded out through the public interface either directly or using NAT. With EZ-NEM, any traffic sent through a secured interface is processed by the crypto policy. If that traffic is defined in the policy, it is encrypted and forwarded on; if not, it's discarded. So when a split-tunnel policy is used, only the secured traffic is processed, and everything else is dropped. Traffic that needs to be forwarded (using say NAT or PAT), but does not qualify against the crypto policy must be processed by a different router interface.

That last point opens the door to another one of the big realities that all engineers face when implementing split-tunneling network-to-network VPNs: When using routers to implement IPsec, you should avoid using the router to process secure and unsecured traffic on the same inside interface. Though it can be done, that does not mean it should be done, and it can easily be avoided. There are three network topology options you should consider:

The concept behind the three options is to have a specific function for each packet handling element such as routing, IPsec packet processing and Internet packet processing. These models do have flexibility in terms of adding or removing hardware elements, but the illustration above captures the essence of the different approaches.

Option 1 is the "one job, one device" approach in which each hardware device handles a specific function. The Internet router provides border security and a filter/monitor on the outside between the location and the ISP. The firewall manages IPsec and user Internet access. This solution is commonly used with GRE tunnels; the job of the remote router is to be the termination point for the GRE tunnel between the location and the core. The firewall then secures this traffic using IPsec. The LAN routing switch has the reachability information for the LAN, core, and core adjacent networks and the default route. The advantages of this solution are that it's great for environments where networking and security are managed in different groups and each functional element is on a different hardware component. The downside is that it's very expensive to implement and requires a lot of coordination to make changes.

Option 2 is the "happy medium" approach. Here the router manages the ISP border, IPsec, and firewall public access, the firewall manages user Internet access, and the LAN routing switch manages IP routing. This approach is far more complex than Option 1, moving the control elements to router interfaces rather than using different hardware devices. However, it still retains the advantage of using a stateful filtering device for managing user Internet traffic. The upside is that there is far less hardware than Option 1. The downside is that it is a more complex design, requiring thoughtful planning and careful maintenance.

Option 3 is the "less is more" approach. The router manages the border, IPsec, and user Internet access. The advantage of the solution is that it's very cost effective. The disadvantage is that the router configuration is again, like Option 2, complex. The IOS supports a protocol-centric stateful and transaction-oriented stateful filtering model, but this function, along with encryption and packet

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

process, is all handled by the router. That makes it essential that you spec the router to handle the appropriate traffic levels, or your users will suffer poor performance.

Our configuration example will support Option 3. A router-based filtering solution is fine for a backup or small remote office connection, where there really is no cost justification for a standalone firewall. Our goal here is to provide secure access to the core and core adjacent networks, and to utilize the local link for Internet access instead of burdening the core with Internet access processing. In terms of core VPN gateway configuration, an additional client policy is required and is included in this VPN gateway configuration example.

You'll also need to create a traffic qualifier ACL to define the IP traffic patterns that will be secured between the VPN gateway and the remote hardware client. The format for this extended ACL is Source=Core Network-->Destination=Remote Network. This ACL requirement does somewhat limit the dynamic flexibility of using the EzVPN client, but depending on the access requirements, "any" can be substituted in the Remote Network field. Also, like in the full-crypto example, Reverse Route Injection (RRI) will enable the core to dynamically announce the remote client networks. RRI is not available on the client side, but there is an alternative we can implement on the client router. We can configure dynamic routing on the remote client router and configure static route redistribution to announce static routes to the core networks utilizing a remote next-hop gateway (coupled with some creative traffic filtering). The configuration of the client router is similar to our previous configuration example, with elements added to support NAT and stateful filtering:

- DHCP/DNS server configuration
- Hardware client configuration
- NAT configuration
- Interface configuration
- IP routing configuration

DHCP/DNS Server Configuration

The configuration of these elements is the same as in the full-crypto hardware client example. One thing to be aware of in this case is that the DNS lookup requests all originate from the router's outside interface directly to the Internet (there is no core DNS server entry). This communication needs to be accounted for when generating the public interface ACLs:

```
outlan-VPN-RTR(config)#ip dhcp pool vlan62
outlan-VPN-RTR(dhcp-config)#network 172.30.62.0 255.255.255.0
outlan-VPN-RTR(dhcp-config)#default-router 172.30.62.1
outlan-VPN-RTR(dhcp-config)#dns-server 1.1.1.1
outlan-VPN-RTR(dhcp-config)#exit
outlan-VPN-RTR(config)#ip dhcp pool vlan89
outlan-VPN-RTR(dhcp-config)#network 172.30.89.0 255.255.255.0
outlan-VPN-RTR(dhcp-config)#default-router 172.30.89.1
outlan-VPN-RTR(dhcp-config)#dns-server 1.1.1.1
outlan-VPN-RTR(dhcp-config)#exit
outlan-VPN-RTR(config)#ip domain name outlan.net
outlan-VPN-RTR(config)#ip name-server 192.36.148.17 192.112.36.4 193.0.
```

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

```
14.129 198.32.64.12
outlan-VPN-RTR(config)#ip dns server
```

Hardware Client Configuration

Again, this step is very simple to the full-crypto example. Here we are using the traffic qualifier ACL to identify the adjacent networks that it will send and receive traffic for:

```
outlan-VPN-RTR(config)#ip access-list extended adj-subnets
outlan-VPN-RTR(config-ext-nacl)# permit ip 172.30.62.0 0.0.0.255 any
outlan-VPN-RTR(config-ext-nacl)# permit ip 172.30.89.0 0.0.0.255 any
outlan-VPN-RTR(config-ext-nacl)#exit
outlan-VPN-RTR(config)#crypto ipsec client ezvpn hard-client
outlan-VPN-RTR(config-crypto-ezvpn)# peer 172.30.80.14
outlan-VPN-RTR(config-crypto-ezvpn)# connect auto
outlan-VPN-RTR(config-crypto-ezvpn)# group hard-client key supersecret
outlan-VPN-RTR(config-crypto-ezvpn)# mode network-extension
outlan-VPN-RTR(config-crypto-ezvpn)# username outlan-rtr1 password outlan-rtr1
outlan-VPN-RTR(config-crypto-ezvpn)# xauth userid mode local
outlan-VPN-RTR(config-crypto-ezvpn)# acl adj-subnets
```

Interface Configuration

The interface configuration for this example is a little more involved than the full-crypto hardware client example due to the addition of reflexive ACLs to provide transaction-based stateful filtering. We start with the public inbound ACL, with permit rules for DHCP and IPsec services. Then we must add support for DNS resolution, ICMP and the evaluate rules for the reflexive lists. Then we create the public outbound list. Cisco's reflexive access lists work by matching traffic with one ACL, then dynamically updating another using an outbound/inbound ACL pair. Static permit or deny rules are inserted above the reflexive match and evaluate rules. On the outbound ACL we need only permit rules for IPsec, followed by the reflexive ACL matching rules. When the outside interface is configured, all that is left are the NAT and EzVPN policy interfaces. Once the EzVPN inside and outside definitions are in place, the client will automatically establish a connection to the VPN gateway.

With the VPN connection up, we still have some IP routing and NAT configuration work to do. But before we configure these elements we must install some protective measures to make sure none of our secured traffic is inadvertently leaked onto the Internet. We'll accomplish this with an extended ACL named "Security-Catch." The job of the catch list is to drop any traffic that should be going to the VPN interface but reaches the NAT interface. Assuming this configuration is being implemented in conjunction with another WAN link, dynamic routing is required to make the LAN routing switch aware of the alternative path to the core networks.

The LAN routing switch must also have a default route. In our example, the default route would be the client router's NAT interface. In the event that the routes to the core networks were missing from the LAN switch's routing table, the LAN switch could forward these routes to the default gateway. The catch filter prevents those routes from being forwarded on and provides the network administrator a means for filtering the Internet services that will be available. Here is the interface configuration:

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

```
outlan-VPN-RTR(config)#ip access-list extended VPN-IN
outlan-VPN-RTR(config-ext-nacl)# remark permit DHCP client Traffic
outlan-VPN-RTR(config-ext-nacl)# permit udp any any eq bootpc
outlan-VPN-RTR(config-ext-nacl)# remark permit IPSEC Phase 1 Phase 2 traffic
outlan-VPN-RTR(config-ext-nacl)# permit esp host 190.55.2.98 any
outlan-VPN-RTR(config-ext-nacl)# permit udp host 190.55.2.98 any eq isakmp
outlan-VPN-RTR(config-ext-nacl)# remark permit local DNS resolution
outlan-VPN-RTR(config-ext-nacl)# permit udp any any gt 1024
outlan-VPN-RTR(config-ext-nacl)# permit udp any any eq domain
outlan-VPN-RTR(config-ext-nacl)# remark allow ICMP
outlan-VPN-RTR(config-ext-nacl)# permit icmp any any
outlan-VPN-RTR(config-ext-nacl)# remark reflexive rules
outlan-VPN-RTR(config-ext-nacl)# evaluate tcp-public-access-temp-list
outlan-VPN-RTR(config-ext-nacl)# evaluate udp-public-access-temp-list
outlan-VPN-RTR(config-ext-nacl)# exit
outlan-VPN-RTR(config)# ip access-list extended VPN-OUT
outlan-VPN-RTR(config-ext-nacl)# remark permit IPSEC Phase 1 Phase 2 traffic
outlan-VPN-RTR(config-ext-nacl)# permit esp any host 190.55.2.98
outlan-VPN-RTR(config-ext-nacl)# permit udp any host 190.55.2.98 eq isakmp
outlan-VPN-RTR(config-ext-nacl)#remark reflexive match rules
outlan-VPN-RTR(config-ext-nacl)# permit tcp any any reflect tcp-public-access-
temp-list
outlan-VPN-RTR(config-ext-nacl)# permit udp any any reflect udp-public-access-
temp-list
outlan-VPN-RTR(config-ext-nacl)#exit
outlan-VPN-RTR(config)# ip access-list extended Security-Catch
outlan-VPN-RTR(config-ext-nacl)#deny ip any 172.30.40.0 0.0.0.255
outlan-VPN-RTR(config-ext-nacl)#deny ip any 172.30.50.0 0.0.0.255
outlan-VPN-RTR(config-ext-nacl)#deny ip any 172.30.60.0 0.0.0.255
outlan-VPN-RTR(config-ext-nacl)#deny ip any 172.30.131.0 0.0.0.255
outlan-VPN-RTR(config-ext-nacl)#deny ip any 172.30.132.0 0.0.0.255
outlan-VPN-RTR(config-ext-nacl)#permit ip any any eq 80
outlan-VPN-RTR(config-ext-nacl)#permit ip any any eq 443
outlan-VPN-RTR(config-ext-nacl)#permit ip any any eq ftp
outlan-VPN-RTR(config-ext-nacl)#permit ip any any eq ftp-data
outlan-VPN-RTR(config-ext-nacl)#permit ip any any established
outlan-VPN-RTR(config)# interface FastEthernet0/0
outlan-VPN-RTR(config-if)#description EzVPN unsecured-public traffic interface
outlan-VPN-RTR(config-if)#ip address dhcp
outlan-VPN-RTR(config-if)#ip access-group VPN-IN in
outlan-VPN-RTR(config-if)#ip access-group VPN-OUT out
outlan-VPN-RTR(config-if)#crypto ipsec client ezvpn hard-client
outlan-VPN-RTR(config-if)#exit
outlan-VPN-RTR(config)# interface Vlan1
outlan-VPN-RTR(config-if)#description EzVPN secured-private traffic interface
outlan-VPN-RTR(config-if)#ip address 172.30.100.1 255.255.255.0
outlan-VPN-RTR(config-if)#crypto ipsec client ezvpn hard-client inside
outlan-VPN-RTR(config-if)#exit
outlan-VPN-RTR(config)# interface FastEthernet0/1
outlan-VPN-RTR(config-if)#description NAT private traffic interface
outlan-VPN-RTR(config-if)#ip address 172.30.120.1 255.255.255.0
```

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

```
outlan-VPN-RTR(config-if)#ip access-group Security-Catch in
```

IP Routing Configuration

As we discussed above, in this example, the LAN routing switch makes the next-hop path decisions for the user nodes. There are two ways to provide this routing data. The simplest way is to install static IP routes on the switch for the core networks, with the hardware client router as the next hop. This approach works fine if the hardware client router is the only gateway with access to the core networks. However, if this solution is being used as a secondary path in conjunction with a leased line or another VPN link connected to the core, we'll need to implement dynamic routing. Because we lack RRI support with the hardware client, we need to get a little creative. To get the core networks to the LAN switch, we can redistribute the static routes. And to ensure that these routes get pulled in the event that the DSL link fails, we can configure them to use a remote next hop (RNH) gateway.

This is only possible because the hardware client will only forward secured traffic when the client Link is established and that link is dependent on the ISP connection in Fa0/0 being active. IP routing is not really in play here; we just need to make sure the traffic that should be secured gets to the inside interface, where it is encapsulated in an ESP IPsec tunnel and sent on to the VPN gateway. It is important that the LAN switch sends the traffic to the correct interface, while it is also aware of any alternative route paths that may be available when this configuration is used as a backup path. The LAN switch will prefer the primary route path; it is only when those routes become unavailable that the LAN switch will send traffic over the RNH.

Configuring the static routes to use an RNH is quite easy. First, define the route for the RNH and tie it to the physical interface of the router. Once the RNH is in place, all subsequent static routers use the RNH as the gateway. Then in the event that the RNH is not available, the static routes tied to it will become invalid and unavailable for route redistribution. It's a great trick that can be used whenever you need to do static redistribution.

After the static routes are in place, we must configure the dynamic routing protocol. For this example, we will use OSPF. Although the router and LAN switch are connected using two different interfaces, only the VPN interface needs to be included in the dynamic protocol configuration. The LAN switch should be configured to use the NAT interface as its static default gateway. Here is our IP routing configuration:

```
outlan-VPN-RTR(config)#ip route 190.55.2.98 255.255.255.255 FastEthernet0/0
outlan-VPN-RTR(config)#ip route 172.30.40.0 255.255.255.0 190.55.2.98
outlan-VPN-RTR(config)#ip route 172.30.50.0 255.255.255.0 190.55.2.98
outlan-VPN-RTR(config)#ip route 172.30.60.0 255.255.255.0 190.55.2.98
outlan-VPN-RTR(config)#ip route 172.30.131.0 255.255.255.0 190.55.2.98
outlan-VPN-RTR(config)#ip route 172.30.132.0 255.255.255.0 190.55.2.98
outlan-VPN-RTR(config)#router ospf 100
outlan-VPN-RTR(config-router)#log-adjacency-changes
outlan-VPN-RTR(config-router)#redistribute static metric 100 subnets
outlan-VPN-RTR(config-router)#network 172.30.100.0 0.0.0.255 area 0.0.0.0
outlan-VPN-RTR(config-router)#router-id 172.30.100.1
```

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

NAT must also be configured, and there is nothing mysterious about the configuration. You must build the traffic match list, define the NAT process, and configure the NAT inside and outside interfaces:

```
outlan-VPN-RTR(config)#access-list 2 permit 172.30.62.0 0.0.0.255
outlan-VPN-RTR(config)#access-list 2 permit 172.30.89.0 0.0.0.255
outlan-VPN-RTR(config)#access-list 2 permit 172.30.120.0 0.0.0.255
outlan-VPN-RTR(config)#ip nat inside source list 2 interface FastEthernet0/0
overload
outlan-VPN-RTR(config)#interface FastEthernet0/1
outlan-VPN-RTR(config-if)#ip nat inside
outlan-VPN-RTR(config-if)#exit
outlan-VPN-RTR(config-if)#interface FastEthernet0/0
outlan-VPN-RTR(config-if)#ip nat outside
outlan-VPN-RTR(config-if)#exit
```

At this point, the client router should be able to process secure and Internet traffic through the correct interfaces (with a little help from the LAN routing switch).

Troubleshooting

It is possible that you may have run into some trouble. Below are some commands that will let you check on the status of the EzVPN client connection and the ISAKMP (Phase 1) and IPsec (Phase 2) security associations. For status on the EzVPN client connection, use the exec command <show crypto ipsec client ezvpn>. This command provides details on what VPN gateway the client is connected to, the backup gateway (if there is one), connection status (IPSEC_ACTIVE means the connection is up) and the inside and outside security interfaces:

```
outlan-VPN-RTR# sh crypto ipsec client ezvpn
Easy VPN Remote Phase: 6
```

```
Tunnel name : hard-client
Inside interface list: Vlan1, Vlan2
Outside interface: FastEthernet0/0
Current State: IPSEC_ACTIVE
Last Event: SOCKET_DOWN
Using PFS Group: 2
Save Password: Allowed
Current EzVPN Peer: 172.30.80.14
Backup gateways
(0): 172.30.80.16
outlan-VPN-RTR#
```

To check on Phase 1 SAs the router has established (or those that have failed or are about to be purged), use the exec command <show crypto isakmp sa>. The word "ACTIVE" in the status column indicates that the ISAKMP SA is valid:

```
outlan-VPN-RTR#sh crypto isakmp sa
IPv4 Crypto ISAKMP SA
```

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN

Michael J. Martin

```
dst      src      state      conn-id slot status
172.30.80.14 172.30.80.225 QM_IDLE      1022  0 ACTIVE
```

outlan-VPN-RTR#

The last crypto status command that really comes in handy provides key information (i.e., transform set, transport mode, number of packets encrypted/decrypted) about the Phase 2 SAs that have been established using the router's crypto map. The exec command <show crypto ipsec SA> lists all of the source (local) destination (remote) subnet pairings for which the router has negotiated security policies. Below is an example for VLAN 100 from our implementation (notice that the destination network is "0.0.0.0/0.0.0.0/0/0" or, in other words, "any."

```
interface: FastEthernet0/0
  Crypto map tag: FastEthernet0/0-head-0, local addr 172.30.80.225

protected vrf: (none)
local ident (addr/mask/prot/port): (172.30.100.1.0/255.255.255.0/0/0)
remote ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
current_peer 172.30.80.14 port 500
  PERMIT, flags={origin_is_acl,}
#pkts encaps: 148833, #pkts encrypt: 148833, #pkts digest: 148833
#pkts decaps: 115407, #pkts decrypt: 115407, #pkts verify: 115407
#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 172.30.80.225, remote crypto endpt.: 172.30.80.14
path mtu 1500, ip mtu 1500, ip mtu idb FastEthernet0/0
current outbound spi: 0xDAD212FE(3671200510)

inbound esp sas:
spi: 0x3D740B43(1031015235)
  transform: esp-3des esp-md5-hmac ,
  in use settings ={Tunnel, }
  conn id: 2261, flow_id: FPGA:261, crypto map: FastEthernet0/0-head-0
  sa timing: remaining key lifetime (k/sec): (4536041/4248)
  IV size: 8 bytes
  replay detection support: Y
  Status: ACTIVE

outbound esp sas:
spi: 0xDAD212FE(3671200510)
  transform: esp-3des esp-md5-hmac ,
  in use settings ={Tunnel, }
  conn id: 2262, flow_id: FPGA:262, crypto map: FastEthernet0/0-head-0
  sa timing: remaining key lifetime (k/sec): (4536973/4248)
  IV size: 8 bytes
  replay detection support: Y
  Status: ACTIVE
```

Split Tunnel VPN Hardware Client Configuration for Cisco EzVPN Michael J. Martin

When implementing any IPsec solution, the `><show crypto isakmp sa>` and `><show crypto ipsec sa>` are essential commands for debugging established peers. In the event that you cannot get two peers to successfully negotiate Phase 1, use the debug command `><debug crypto isakmp>`. The command produces copious amounts of data on the IKE negotiation and will reveal any issues or incompatibilities.