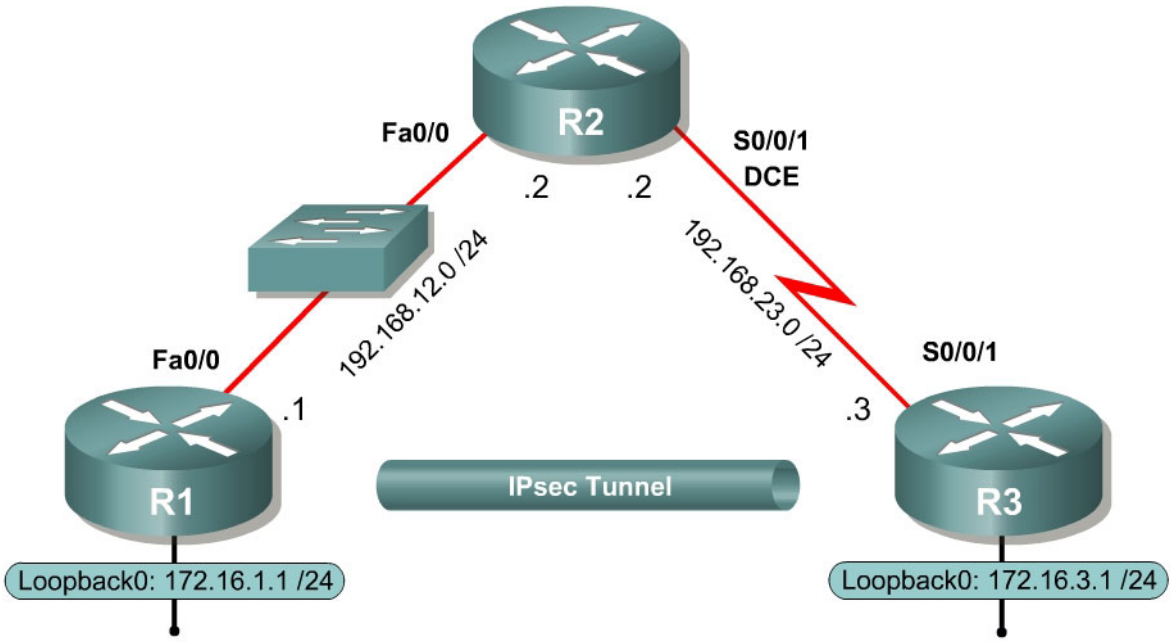
Lab 3.5 Configuring Site-to-Site IPsec VPNs with the IOS CLI

Learning Objectives

- Configure EIGRP on the routers
- Create a site-to-site IPsec VPN using IOS
- Verify IPsec operation

Topology Diagram



Scenario

In this lab, you will configure a site-to-site IPsec VPN. Once you have configured the VPN, the traffic between the loopback interfaces on R1 and R3 will be encrypted.

You will use the Cisco IOS command-line interface (CLI) for this lab exercise. Lab 3.4 involves the same function as this exercise, but implemented via the Cisco Security Device Manager (SDM).

Step 1: Configure Addressing

Configure the loopback interfaces and the serial interfaces with the addresses shown in the diagram. Set the clock rates on the appropriate interfaces and issue the **no shutdown** command on all physical connections. Verify that you have connectivity across local subnets using the **ping** command.


```
R1(config)# interface loopback0
R1(config-if)# ip address 172.16.1.1 255.255.255.0
R1(config-if)# interface fastethernet0/0
R1(config-if)# ip address 192.168.12.1 255.255.255.0
R1(config-if)# no shutdown

R2(config)# interface fastethernet0/0
R2(config-if)# ip address 192.168.12.2 255.255.255.0
R2(config-if)# no shutdown
R2(config-if)# interface serial0/0/1
R2(config-if)# ip address 192.168.23.2 255.255.255.0
R2(config-if)# clockrate 64000
R2(config-if)# no shutdown

R3(config)# interface loopback0
R3(config-if)# ip address 172.16.3.1 255.255.255.0
R3(config-if)# interface serial0/0/1
R3(config-if)# ip address 192.168.23.3 255.255.255.0
R3(config-if)# no shutdown
```

Step 2: Configure EIGRP

In order to maintain connectivity between remote networks, configure EIGRP to route between all networks in the diagram. Add all connected subnets into the EIGRP autonomous system on every router. Disable automatic summarization.

```
R1(config)# router eigrp 1
R1(config-router)# no auto-summary
R1(config-router)# network 172.16.0.0
R1(config-router)# network 192.168.12.0

R2(config)# router eigrp 1
R2(config-router)# no auto-summary
R2(config-router)# network 192.168.12.0
R2(config-router)# network 192.168.23.0

R3(config)# router eigrp 1
R3(config-router)# no auto-summary
R3(config-router)# network 172.16.0.0
R3(config-router)# network 192.168.23.0
```

Verify that you have full IP connectivity at this point using the following TCL script.

```
tclsh

foreach address {
172.16.1.1
192.168.12.1
192.168.12.2
192.168.23.2
172.16.3.1
192.168.23.3
} { ping $address }

tclquit
```

Compare your output with the output shown in Appendix A. Troubleshoot as necessary.

Step 3: Create IKE Policies

IPsec is a framework of open standards developed by the Internet Engineering Task Force (IETF). It provides security for transmission of sensitive information over unprotected networks such as the Internet. IPsec acts at the network layer, protecting and authenticating IP packets between participating IPsec devices (peers), such as Cisco routers.

Since IPSec is a framework, it allows us to exchange security protocols as new technologies (including encryption algorithms) are developed.

There are two central configuration elements to the implementation of an IPsec VPN:

1. Implement Internet Key Exchange (IKE) parameters
2. Implement IPsec parameters

The exchange method employed by IKE is first used to pass and validate IKE policies between peers. Then, the peers exchange and match IPSec policies for the authentication and encryption of data traffic. The IKE policy controls the authentication, encryption algorithm, and key exchange method used for IKE proposals that are sent and received by the IPSec endpoints. The IPSec policy is used to encrypt data traffic sent through the VPN tunnel.

IKE will need to be enabled for IPsec to work. IKE is enabled by default on IOS images with cryptographic feature sets. If it is disabled for some reason, you can enable it with the command **crypto isakmp enable**.

```
R1(config)# crypto isakmp enable
```

If you cannot execute this command on the router, you need to upgrade the IOS image to an image with a feature set that includes the Cisco cryptographic services.

The exchange method employed by IKE is first used to pass and validate IKE policies between peers. Then, the peers exchange and match IPsec policies for the authentication and encryption of data traffic. The IKE policy controls the authentication, encryption algorithm, and key exchange method that is used by IKE proposals that are sent and received by the IPsec endpoints. The IPsec policy is used to encrypt data traffic that is sent through the VPN tunnel.

To allow IKE Phase 1 negotiation, you must create an Internet Security Association and Key Management Protocol (ISAKMP) policy and configure a peer association involving that ISAKMP policy. An ISAKMP policy defines the authentication and encryption algorithms and hash function used to send control traffic between the two VPN endpoints. When an ISAKMP security

association has been accepted by the IKE peers, IKE Phase 1 has been completed. IKE Phase 2 parameters will be configured later in the lab.

Issue the **crypto isakmp policy** *number* command in global configuration mode. This initiates the ISAKMP policy configuration mode. Once in this mode, you can view the various IKE parameters available by typing **?**. Enter into this configuration mode on R1 for policy 10, and view some of the possible settings.

```
R1(config)# crypto isakmp policy 10
R1(config-isakmp)# ?
ISAKMP commands:
  authentication  Set authentication method for protection suite
  default         Set a command to its defaults
  encryption      Set encryption algorithm for protection suite
  exit            Exit from ISAKMP protection suite configuration mode
  group           Set the Diffie-Hellman group
  hash            Set hash algorithm for protection suite
  lifetime        Set lifetime for ISAKMP security association
  no              Negate a command or set its defaults
```

Your choice of an encryption algorithm controls how confidential the control channel between the endpoints will be. The hash algorithm controls data integrity, that is, surety that the data received from a peer has not been tampered with in transit. The authentication type ensures that the packet was indeed sent and signed by the remote peer. The Diffie-Hellman group is used to create a secret key shared by the peers that has never been sent across the network.

Configure an authentication type of pre-shared keys. Use AES 256 encryption, SHA as your hash algorithm, and Diffie-Hellman group 5 for this IKE policy.

Give the policy a lifetime of 3600 seconds (one hour). Configure the same policy on R3. Older versions of the IOS do not support AES 256 encryption and/or SHA as your hash algorithm. Substitute whatever encryption and hashing algorithm your router supports. Be sure the same changes are made on the other VPN endpoint so that they are the same.

```
R1(config)# crypto isakmp policy 10
R1(config-isakmp)# authentication pre-share
R1(config-isakmp)# encryption aes 256
R1(config-isakmp)# hash sha
R1(config-isakmp)# group 5
R1(config-isakmp)# lifetime 3600

R3(config)# crypto isakmp policy 10
R3(config-isakmp)# authentication pre-share
R3(config-isakmp)# encryption aes 256
R3(config-isakmp)# hash sha
R3(config-isakmp)# group 5
R3(config-isakmp)# lifetime 3600
```

Although you only need to configure one policy here, you can configure multiple IKE policies. The different priority numbers refer to how secure a policy is. The lower the policy number is, the more secure a policy is. Routers will check to

verify which security policies are compatible with their peer, starting with the lowest numbered (most secure) policies. You can verify your IKE policy with the **show crypto isakmp policy** command. Note that a default, less secure policy already exists on the router.

```
R1# show crypto isakmp policy

Global IKE policy
Protection suite of priority 10
        encryption algorithm:   AES - Advanced Encryption Standard (256 bit
keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #5 (1536 bit)
        lifetime:               3600 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit

R3# show crypto isakmp policy

Global IKE policy
Protection suite of priority 10
        encryption algorithm:   AES - Advanced Encryption Standard (256 bit
keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Pre-Shared Key
        Diffie-Hellman group:   #5 (1536 bit)
        lifetime:               3600 seconds, no volume limit
Default protection suite
        encryption algorithm:   DES - Data Encryption Standard (56 bit keys).
        hash algorithm:         Secure Hash Standard
        authentication method:  Rivest-Shamir-Adleman Signature
        Diffie-Hellman group:   #1 (768 bit)
        lifetime:               86400 seconds, no volume limit
```

Step 4: Configure Pre-Shared Keys

Since we chose pre-shared keys as our authentication method in the IKE policy, we must configure a key on each router corresponding to the other VPN endpoint. These keys must match up for authentication to be successful and for the IKE peering to be completed. For simplicity we can use the key “cisco”. A production network should use a more complex key.

Use the global configuration command **crypto isakmp key** *key-string* **address** *address* to enter a pre-shared key. Use the IP address of the remote peer. Ensure that the IP address is the remote interface that the peer would use to route traffic to the local router.

Which IP addresses should you use to configure the IKE peers, given the topology diagram?

Each IP address that is used to configure the IKE peers are also referred to as the IP address of the remote VPN endpoint. You can also specify the peer by hostname (substitute the keyword **address** with **hostname**) if the IP address may change a lot. You either have to statically bind the IP address to the hostname on the router, or use a name-lookup service.

```
R1(config)# crypto isakmp key cisco address 192.168.23.3

R3(config)# crypto isakmp key cisco address 192.168.12.1
```

Step 5: Configure the IPsec Transform Set and Lifetimes

The IPsec transform set is another crypto configuration parameter that routers negotiate to form a security association. In the same way that ISAKMP policies can, multiple transform sets can exist on a router. Routers will compare their transform sets to the remote peer until they find a transform set that matches exactly.

Create an IPsec transform set, using the syntax **crypto ipsec transform-set** *tag parameters*. Use **?** to see what parameters are available. For R1 and R3, create a transform set with tag 50 and use an ESP transform with an AES 256 cipher first, with Encapsulating Security Protocol (ESP) and the SHA hash function and finally an authentication header using SHA.

```
R1(config)#crypto ipsec transform-set ?
  WORD  Transform set tag

R1(config)#crypto ipsec transform-set 50 ?
  ah-md5-hmac   AH-HMAC-MD5 transform
  ah-sha-hmac   AH-HMAC-SHA transform
  comp-lzs      IP Compression using the LZS compression algorithm
  esp-3des      ESP transform using 3DES(EDE) cipher (168 bits)
  esp-aes       ESP transform using AES cipher
  esp-des       ESP transform using DES cipher (56 bits)
  esp-md5-hmac  ESP transform using HMAC-MD5 auth
  esp-null      ESP transform w/o cipher
  esp-seal      ESP transform using SEAL cipher (160 bits)
  esp-sha-hmac  ESP transform using HMAC-SHA auth

R1(config)#crypto ipsec transform-set 50  esp-aes ?
  128           128 bit keys.
  192           192 bit keys.
  256           256 bit keys.
  ah-md5-hmac   AH-HMAC-MD5 transform
  ah-sha-hmac   AH-HMAC-SHA transform
  comp-lzs      IP Compression using the LZS compression algorithm
  esp-md5-hmac  ESP transform using HMAC-MD5 auth
  esp-sha-hmac  ESP transform using HMAC-SHA auth
  <cr>

R1(config)#crypto ipsec transform-set 50  esp-aes 256 ?
  ah-md5-hmac   AH-HMAC-MD5 transform
  ah-sha-hmac   AH-HMAC-SHA transform
```



```
  comp-lzs      IP Compression using the LZS compression algorithm
  esp-md5-hmac  ESP transform using HMAC-MD5 auth
  esp-sha-hmac  ESP transform using HMAC-SHA auth
  <cr>

R1(config)#crypto ipsec transform-set 50  esp-aes 256 esp-sha-hmac ?
  ah-md5-hmac  AH-HMAC-MD5 transform
  ah-sha-hmac  AH-HMAC-SHA transform
  comp-lzs     IP Compression using the LZS compression algorithm
  <cr>
```

Executing the above command sends you into the transform set configuration mode, although you can just type **exit** to leave it since you do not need to configure any additional transform parameters.

```
R1(config)# crypto ipsec transform-set 50 esp-aes 256 esp-sha-hmac ah-sha-hmac
R1(cfg-crypto-trans)# exit
R1(config)#

R3(config)# crypto ipsec transform-set 50 esp-aes 256 esp-sha-hmac ah-sha-hmac
R3(cfg-crypto-trans)# exit
R3(config)#
```

What is the function of the IPsec transform set?

You can also change the IPsec security association lifetimes from its default which is 3600 seconds or 4,608,000 kilobytes, whichever comes first. Change this with the global configuration command **crypto ipsec security-association lifetime seconds** *seconds* or **crypto ipsec security-association lifetime kilobytes** *kilobytes*. On R1 and R3, set the IPsec security association lifetime to 30 minutes, or 1800 seconds.

```
R1(config)# crypto ipsec security-association lifetime seconds 1800

R3(config)# crypto ipsec security-association lifetime seconds 1800
```

Step 6: Define Interesting Traffic

Now that most of the encryption settings are in place, define extended access lists to tell the router which traffic to encrypt. Like other access lists used to define “interesting traffic” rather than packet filtering, permit and deny do not have the usual meaning of a filtering access list. A packet which is permitted by an access list used for defining IPsec traffic will get encrypted if the IPsec session is configured correctly. A packet that is denied by one of these access lists will not be dropped; it will be sent unencrypted. Also, like any other access list, there is an implicit **deny** at the end, which in this case means the default action is to not encrypt traffic. If there is no IPsec security association correctly

configured, then no traffic will be encrypted, but traffic will be forwarded as unencrypted traffic.

In this scenario, the traffic you want to encrypt is traffic going from R1's loopback network to R3's loopback network, or vice versa. These access lists are used outbound on the VPN endpoint interfaces, so configure them accordingly. The configuration of R1's access list will need to be mirrored exactly on R3 for this to work properly.

```
R1(config)# access-list 101 permit ip 172.16.1.0 0.0.0.255 172.16.3.0
0.0.0.255

R3(config)# access-list 101 permit ip 172.16.3.0 0.0.0.255 172.16.1.0
0.0.0.255
```

Does IPsec evaluate whether the access lists are mirrored as a requirement to negotiate its security association?

Step 7: Create and Apply Crypto Maps

Now that you have created all of these small configuration modules, you can bring them together in a crypto map. A crypto map is a mapping that associates traffic matching an access list (like the one we created earlier) to a peer and various IKE and IPsec settings. Crypto maps can have multiple map statements, so you can have traffic that matches a certain access list being encrypted and sent to one IPsec peer, and have other traffic that matches a different access list being encrypted towards a different peer. After a crypto map is created, it can be applied to one or more interfaces. The interface(s) that it is applied to should be the one(s) facing the IPsec peer.

To create a crypto map, use the global configuration command **crypto map** *name sequence-num type* to enter the crypto map configuration mode for that sequence number. Multiple crypto map statements can belong to the same crypto map, and they will be evaluated in ascending numerical order.

Use a type of **ipsec-isakmp**, which means IKE will be used to establish IPsec security associations. Under normal circumstances, you would want to use this mode, as opposed to the **ipsec-manual** type. If ipsec-manual is used, IKE will not be used to configure the IPsec security association (SA). (This mode is beyond the scope of this lab.) Name the crypto map "MYMAP," and use 10 as the sequence number. Enter the crypto map configuration mode on R1. When you do this, the crypto map will be created and this command will generate a

warning that a peer must be fully configured before the crypto map is considered valid and can be actively applied.

```
R1(config)# crypto map MYMAP 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
```

Use the **match address** *access-list* command to specify which access list will define which traffic to encrypt. If you have ever configured route maps or similar maps on a router before, some of these map-related commands may seem familiar.

```
R1(config-crypto-map)# match address 101
```

There are many possible **set** commands that you can do in a crypto map. To view the list of possibilities, use the **?** help character.

```
R1(config-crypto-map)# set ?
  identity              Identity restriction.
  ip                    Interface Internet Protocol config commands
  isakmp-profile        Specify isakmp Profile
  nat                   Set NAT translation
  peer                  Allowed Encryption/Decryption peer.
  pfs                   Specify pfs settings
  security-association  Security association parameters
  transform-set         Specify list of transform sets in priority order
```

Setting a peer IP or hostname is required, so set it to R3's remote VPN endpoint interface using the **set peer** *address* command. Hardcode the transform set to be used with this peer, using the **set transform-set** *tag* command.

Set the perfect forwarding secrecy type using the **set pfs** *type* command, and also modify the default IPsec security association lifetime with the **set security-association lifetime seconds** *seconds* command. As you can see in the output of the **?** above, there are more settings you can change in this crypto map. Create a matching crypto map on R3 using the mirrored access list to define interesting traffic.

```
R1(config-crypto-map)# set peer 192.168.23.3
R1(config-crypto-map)# set pfs group5
R1(config-crypto-map)# set transform-set 50
R1(config-crypto-map)# set security-association lifetime seconds 900

R3(config)# crypto map MYMAP 10 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
R3(config-crypto-map)# match address 101
R3(config-crypto-map)# set peer 192.168.12.1
R3(config-crypto-map)# set pfs group5
R3(config-crypto-map)# set transform-set 50
R3(config-crypto-map)# set security-association lifetime seconds 900
```

Now that the crypto maps are created, the last step in the process of creating site-to-site VPNs is applying the maps to interfaces. This is done with the

interface level **crypto map** *name* command. Note that the SAs will not be established until the crypto map has been activated by interesting traffic. Do not create interesting traffic yet, because you will want to enable some debugging during the next step. The router will generate a notification that crypto is now on.

```
R1(config)# interface fastethernet0/0
R1(config-if)# crypto map MYMAP
*Jan 17 04:09:09.150: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON

R3(config)# interface serial0/0/1
R3(config-if)# crypto map MYMAP
*Jan 17 04:10:54.138: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
```

Step 8: Verify IPsec Configuration

In Step 3, you used the **show crypto isakmp policy** command to show the configured ISAKMP policies on the router. Similarly, the **show crypto ipsec transform-set** command displays the configured IPsec policies in the form of the transport sets.

```
R1# show crypto ipsec transform-set
Transform set 50: { ah-sha-hmac  }
   will negotiate = { Tunnel,  },
   { esp-256-aes esp-sha-hmac  }
   will negotiate = { Tunnel,  },

R3# show crypto ipsec transform-set
Transform set 50: { ah-sha-hmac  }
   will negotiate = { Tunnel,  },
   { esp-256-aes esp-sha-hmac  }
   will negotiate = { Tunnel,  },
```

Use the **show crypto map** command to display the crypto maps that will be applied to the router.

```
R1# show crypto map
Crypto Map "MYMAP" 10 ipsec-isakmp
        Peer = 192.168.23.3
        Extended IP access list 101
           access-list 101 permit ip 172.16.1.0 0.0.0.255 172.16.3.0 0.0.0.255
        Current peer: 192.168.23.3
        Security association lifetime: 4608000 kilobytes/900 seconds
        PFS (Y/N): Y
        DH group:  group5
        Transform sets={
                50,
        }
        Interfaces using crypto map MYMAP:
                FastEthernet0/0

R3# show crypto map
Crypto Map "MYMAP" 10 ipsec-isakmp
        Peer = 192.168.12.1
        Extended IP access list 101
           access-list 101 permit ip 172.16.3.0 0.0.0.255 172.16.1.0 0.0.0.255
        Current peer: 192.168.12.1
        Security association lifetime: 4608000 kilobytes/900 seconds
```



```
        PFS (Y/N): Y
        DH group:  group5
        Transform sets={
                50,
        }
        Interfaces using crypto map MYMAP:
                Serial0/0/1
```

The output of these **show** commands will not change if interesting traffic goes across the connection.

Step 9: Verify IPsec Operation

If you use the **show crypto isakmp sa** command, it will reveal that no IKE SAs exist yet. Once we send some interesting traffic later in this lab, this command output will change.

```
R1# show crypto isakmp sa
dst             src             state          conn-id slot status

R3# show crypto isakmp sa
dst             src             state          conn-id slot status
```

If you use the **show crypto ipsec sa** command, this command will show the unused SA between R1 and R3. Note the number of packets sent across and the lack of any security associations listed towards the bottom of the output.

```
R1# show crypto ipsec sa

interface: FastEthernet0/0
    Crypto map tag: MYMAP, local addr 192.168.12.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.16.1.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (172.16.3.0/255.255.255.0/0/0)
   current_peer 192.168.23.3 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #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.: 192.168.12.1, remote crypto endpt.: 192.168.23.3
     path mtu 1500, ip mtu 1500, ip mtu idb FastEthernet0/0
     current outbound spi: 0x0(0)

     inbound esp sas:

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:

     outbound ah sas:

     outbound pcp sas:
```



```
R3# show crypto ipsec sa

interface: Serial0/0/1
    Crypto map tag: MYMAP, local addr 192.168.23.3

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.16.3.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (172.16.1.0/255.255.255.0/0/0)
   current_peer 192.168.12.1 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0
    #pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
    #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.: 192.168.23.3, remote crypto endpt.: 192.168.12.1
     path mtu 1500, ip mtu 1500, ip mtu idb Serial0/0/1
     current outbound spi: 0x0(0)

     inbound esp sas:

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:

     outbound ah sas:

     outbound pcp sas:
```

Why have no security associations (SAs) been negotiated?

How could you force the IPsec peers to negotiate their security association?

Step 10: Interpret IPsec Event Debugging

In terms of the actual communication between the VPN endpoints, ISAKMP prescribes stringent rules as to how an SA can be established. IKE Phase I (ISAKMP) will negotiate the secure channel between the endpoints, authenticate the neighbor as having the correct secret key, and authenticate the remote endpoint through the secure channel. IKE Phase I will use “main mode,” which consists of six messages in three event-driven exchanges. The result is one bidirectional ISAKMP security association. The exchanges are input/output-

driven, so every event is recorded in the debug as an input event from either the local router or the remote router.

IKE Phase II (IPsec) will negotiate the IPsec tunnel between the two endpoints, authenticate the peers, and encrypt data traffic between them through the encrypted tunnel. IKE Phase II uses the process called “quick mode” to perform its exchange to establish two unidirectional security associations.

On R1, enable two debug commands: **debug crypto isakmp** and **debug crypto ipsec**.

```
R1# debug crypto isakmp
Crypto ISAKMP debugging is on
R1# debug crypto ipsec
Crypto IPSEC debugging is on
```

Now, send an extended ping from R1’s loopback to R3’s loopback, and watch the debug outputs on both routers. You will see both ISAKMP negotiation and IPsec SAs being established. This output is *very* verbose.

```
R1# ping
Protocol [ip]:
Target IP address: 172.16.3.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
```

At this point, the packets will be sent. What will happen next?

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
Packet sent with a source address of 172.16.1.1

*Jan 17 05:11:39.142: IPSEC(sa_request): ,
  (key eng. msg.) OUTBOUND local= 192.168.12.1, remote= 192.168.23.3,
    local_proxy= 172.16.1.0/255.255.255.0/0/0 (type=4),
    remote_proxy= 172.16.3.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= NONE  (Tunnel),
    lifedur= 900s and 4608000kb,
    spi= 0x0(0), conn_id= 0, keysize= 256, flags= 0x0
...
```

When R1 detects interesting traffic going toward R3, the **crypto map** statement defined on R1’s Fast Ethernet interface invokes an IPsec state change from inactive to active. The IPsec suite attempts to raise a security association between R1 and R3 to pass secure traffic with the IPsec parameters that were previously configured. The ISAKMP processes on each of the VPN endpoints

are now aware that a security association will be attempted and prepare to send ISAKMP policies.

```
...
*Jan 17 05:11:39.146: ISAKMP:(0): SA request profile is (NULL)
*Jan 17 05:11:39.146: ISAKMP: Created a peer struct for 192.168.23.3, peer
port 500
*Jan 17 05:11:39.146: ISAKMP: New peer created peer = 0x46F56220 peer_handle =
0x80000002
*Jan 17 05:11:39.146: ISAKMP: Locking peer struct 0x46F56220, refcount 1 for
isakmp_initiator
*Jan 17 05:11:39.146: ISAKMP: local port 500, remote port 500
*Jan 17 05:11:39.146: ISAKMP: set new node 0 to QM_IDLE
*Jan 17 05:11:39.146: insert sa successfully sa = 477B9850
...
```

The Cisco IOS software builds an ISAKMP peer structure in memory, providing a means to store parameters and policies related to IKE Phase I key exchanges. The peers will communicate on port 500 on both ends between the IP addresses of the encrypting interfaces on each router. ISAKMP creates and inserts a memory structure representing the ISAKMP security association into the peer structure it just created.

```
...
*Jan 17 05:11:39.146: ISAKMP:(0):Can not start Aggressive mode, trying Main
mode
*Jan 17 05:11:39.146: ISAKMP:(0):found peer pre-shared key matching
192.168.23.3
*Jan 17 05:11:39.146: ISAKMP:(0): constructed NAT-T vendor-07 ID
*Jan 17 05:11:39.150: ISAKMP:(0): constructed NAT-T vendor-03 ID
*Jan 17 05:11:39.150: ISAKMP:(0): constructed NAT-T vendor-02 ID
*Jan 17 05:11:39.150: ISAKMP:(0):Input = IKE_MESG_FROM_IPSEC, IKE_SA_REQ_MM
*Jan 17 05:11:39.150: ISAKMP:(0):Old State = IKE_READY  New State = IKE_I_MM1
...
```

Aggressive mode is an exchange process in which all of IKE Phase I is negotiated with one exchange. This aggressive mode is clearly less secure than main mode, which relies on three exchanges: the ISAKMP policy exchange, the Diffie-Hellman key exchange, and an encrypted authentication test that initiates the ISAKMP security association used for Phase II. In main mode, less information is given to the remote node before the remote node must communicate and can be authenticated.

```
...
*Jan 17 05:11:39.150: ISAKMP:(0): beginning Main Mode exchange
*Jan 17 05:11:39.150: ISAKMP:(0): sending packet to 192.168.23.3 my_port 500
peer_port 500 (I) MM_NO_STATE
*Jan 17 05:11:39.282: ISAKMP (0:0): received packet from 192.168.23.3 dport
500 sport 500 Global (I) MM_NO_STATE
*Jan 17 05:11:39.286: ISAKMP:(0):Input = IKE_MESG_FROM_PEER, IKE_MM_EXCH
*Jan 17 05:11:39.286: ISAKMP:(0):Old State = IKE_I_MM1  New State = IKE_I_MM2

*Jan 17 05:11:39.286: ISAKMP:(0): processing SA payload. message ID = 0

.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 52/54/56 ms
```



```
R1#

*Jan 17 05:11:39.286: ISAKMP:(0): processing vendor id payload
*Jan 17 05:11:39.286: ISAKMP:(0): vendor ID seems Unity/DPD but major 245
mismatch
*Jan 17 05:11:39.286: ISAKMP (0:0): vendor ID is NAT-T v7
*Jan 17 05:11:39.286: ISAKMP:(0):found peer pre-shared key matching
192.168.23.3
*Jan 17 05:11:39.286: ISAKMP:(0): local preshared key found
*Jan 17 05:11:39.286: ISAKMP : Scanning profiles for xauth ...
*Jan 17 05:11:39.286: ISAKMP:(0):Checking ISAKMP transform 1 against priority
10 policy
*Jan 17 05:11:39.286: ISAKMP:      encryption AES-CBC
*Jan 17 05:11:39.286: ISAKMP:      keylength of 256
*Jan 17 05:11:39.286: ISAKMP:      hash SHA
*Jan 17 05:11:39.286: ISAKMP:      default group 5
*Jan 17 05:11:39.286: ISAKMP:      auth pre-share
*Jan 17 05:11:39.286: ISAKMP:      life type in seconds
*Jan 17 05:11:39.286: ISAKMP:      life duration (basic) of 3600
*Jan 17 05:11:39.286: ISAKMP:(0):atts are acceptable. Next payload is 0
*Jan 17 05:11:39.286: ISAKMP:(0): processing vendor id payload
*Jan 17 05:11:39.286: ISAKMP:(0): vendor ID seems Unity/DPD but major 245
mismatch
*Jan 17 05:11:39.286: ISAKMP (0:0): vendor ID is NAT-T v7
*Jan 17 05:11:39.290: ISAKMP:(0):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_MAIN_MODE
*Jan 17 05:11:39.290: ISAKMP:(0):Old State = IKE_I_MM2  New State = IKE_I_MM2
...
```

During the first exchange, the initiator sends its policy to the endpoint of the ISAKMP security association in the first message and the endpoint responds with its ISAKMP security policy in the second message.

Notice that each of the IKE peers sends the ISAKMP security policy to the other. R1 then processes the payload of the packet it received from R3 and determines that it has a pre-shared key associated with R3's address. R1 matches the ISAKMP transform set (policy) from R3 against its own priority 10 policy. After this check is performed, R1 determines that the attributes of the ISAKMP policy are acceptable and signals the ISAKMP process to continue to the next main mode exchange. Finally, R1 informs R3 that it has accepted the policy and entered the second exchange by beginning the second exchange with R3.

Also, note that the Internet Control Message Protocol (ICMP) packets passed through the tunnel although the debug output has not finished displaying to the screen. The entire exchange took about one second to perform, according to the timestamps on the debug messages.

```
...
*Jan 17 05:11:39.290: ISAKMP:(0): sending packet to 192.168.23.3 my_port 500
peer_port 500 (I) MM_SA_SETUP
*Jan 17 05:11:39.290: ISAKMP:(0):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_COMPLETE
*Jan 17 05:11:39.290: ISAKMP:(0):Old State = IKE_I_MM2  New State = IKE_I_MM3
*Jan 17 05:11:39.502: ISAKMP (0:0): received packet from 192.168.23.3 dport
500 sport 500 Global (I) MM_SA_SETUP
*Jan 17 05:11:39.502: ISAKMP:(0):Input = IKE_MESG_FROM_PEER, IKE_MM_EXCH
```



```
*Jan 17 05:11:39.502: ISAKMP:(0):Old State = IKE_I_MM3  New State = IKE_I_MM4
*Jan 17 05:11:39.506: ISAKMP:(0): processing KE payload. message ID = 0
*Jan 17 05:11:39.638: ISAKMP:(0): processing NONCE payload. message ID = 0
*Jan 17 05:11:39.638: ISAKMP:(0):found peer pre-shared key matching
192.168.23.3
*Jan 17 05:11:39.642: ISAKMP:(1001): processing vendor id payload
*Jan 17 05:11:39.642: ISAKMP:(1001): vendor ID is Unity
*Jan 17 05:11:39.642: ISAKMP:(1001): processing vendor id payload
*Jan 17 05:11:39.642: ISAKMP:(1001): vendor ID is DPD
*Jan 17 05:11:39.642: ISAKMP:(1001): processing vendor id payload
*Jan 17 05:11:39.642: ISAKMP:(1001): speaking to another IOS box!
*Jan 17 05:11:39.642: ISAKMP:(1001):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_MAIN_MODE
*Jan 17 05:11:39.642: ISAKMP:(1001):Old State = IKE_I_MM4  New State =
IKE_I_MM4
...
```

During main mode's second exchange, the ISAKMP initiator sends the remote endpoint an RSA nonce (random number), to be used in the Diffie-Hellman algorithm, and the key it receives as the output of the Diffie-Hellman function. This is the third message in the main mode process. The remote endpoint R3 then replies in the fourth message of main mode with its respective nonce and key. R1 and R3 authenticate each other as both sharing the same pre-shared key used to generate the key it received from the peer.

At this point, R1 prepares to send a packet to R3 that will be passed through the secure channel. This packet will be used by R3 to authenticate R1 as the remote end of the ISAKMP security association.

```
...
*Jan 17 05:11:39.642: ISAKMP:(1001):Send initial contact
*Jan 17 05:11:39.646: ISAKMP:(1001):SA is doing pre-shared key authentication
using id type ID_IPV4_ADDR
*Jan 17 05:11:39.646: ISAKMP (0:1001): ID payload
        next-payload : 8
        type         : 1
        address      : 192.168.12.1
        protocol     : 17
        port         : 500
        length       : 12
*Jan 17 05:11:39.646: ISAKMP:(1001):Total payload length: 12
*Jan 17 05:11:39.646: ISAKMP:(1001): sending packet to 192.168.23.3 my_port
500 peer_port 500 (I) MM_KEY_EXCH
*Jan 17 05:11:39.646: ISAKMP:(1001):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_COMPLETE
*Jan 17 05:11:39.646: ISAKMP:(1001):Old State = IKE_I_MM4  New State =
IKE_I_MM5
*Jan 17 05:11:39.690: ISAKMP (0:1001): received packet from 192.168.23.3 dport
500 sport 500 Global (I) MM_KEY_EXCH
*Jan 17 05:11:39.690: ISAKMP:(1001): processing ID payload. message ID = 0
*Jan 17 05:11:39.690: ISAKMP (0:1001): ID payload
        next-payload : 8
        type         : 1
        address      : 192.168.23.3
        protocol     : 17
        port         : 500
        length       : 12
*Jan 17 05:11:39.690: ISAKMP:(0):: peer matches *none* of the profiles
*Jan 17 05:11:39.690: ISAKMP:(1001): processing HASH payload. message ID = 0
```



```
*Jan 17 05:11:39.690: ISAKMP:(1001):SA authentication status:
        authenticated
*Jan 17 05:11:39.690: ISAKMP:(1001):SA has been authenticated with
192.168.23.3
...
```

When R1 receives the authentication packet from R3, it checks the ID payload and hash value of the packet. If ISAKMP authenticates the security association through the encrypted channel, IKE Phase I is complete and the security association has been established.

```
...
*Jan 17 05:11:39.690: ISAKMP: Trying to insert a peer
192.168.12.1/192.168.23.3/500/,  and inserted successfully 46F56220.
*Jan 17 05:11:39.690: ISAKMP:(1001):Input = IKE_MESG_FROM_PEER, IKE_MM_EXCH
*Jan 17 05:11:39.690: ISAKMP:(1001):Old State = IKE_I_MM5  New State =
IKE_I_MM6
*Jan 17 05:11:39.694: ISAKMP:(1001):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_MAIN_MODE
*Jan 17 05:11:39.694: ISAKMP:(1001):Old State = IKE_I_MM6  New State =
IKE_I_MM6
*Jan 17 05:11:39.694: ISAKMP:(1001):Input = IKE_MESG_INTERNAL,
IKE_PROCESS_COMPLETE
*Jan 17 05:11:39.694: ISAKMP:(1001):Old State = IKE_I_MM6  New State =
IKE_P1_COMPLETE
...
```

Finally, R1 inserts the peer into the memory structure reserved at the beginning of the ISAKMP negotiation. R1 begins IKE Phase II over the ISAKMP security association created in IKE Phase I.

```
...
*Jan 17 05:11:39.694: ISAKMP:(1001):beginning Quick Mode exchange, M-ID of
787769575
*Jan 17 05:11:39.694: ISAKMP:(1001):QM Initiator gets spi
*Jan 17 05:11:39.698: ISAKMP:(1001): sending packet to 192.168.23.3 my_port
500 peer_port 500 (I) QM_IDLE
*Jan 17 05:11:39.698: ISAKMP:(1001):Node 787769575, Input = IKE_MESG_INTERNAL,
IKE_INIT_QM
*Jan 17 05:11:39.698: ISAKMP:(1001):Old State = IKE_QM_READY  New State =
IKE_QM_I_QM1
*Jan 17 05:11:39.698: ISAKMP:(1001):Input = IKE_MESG_INTERNAL,
IKE_PHASE1_COMPLETE
*Jan 17 05:11:39.698: ISAKMP:(1001):Old State = IKE_P1_COMPLETE  New State =
IKE_P1_COMPLETE
...
```

Quick mode uses three messages to create an IPsec security association. The initiator, R1, sends the first message including the hash, IPsec security association policies, a nonce, and a key created using the pre-shared keys and two ID payloads. R3 processes R1's IPsec proposal and replies with a message with its values for the above parameters.

```
...
*Jan 17 05:11:40.014: ISAKMP (0:1001): received packet from 192.168.23.3 dport
500 sport 500 Global (I) QM_IDLE
*Jan 17 05:11:40.018: ISAKMP:(1001): processing HASH payload. message ID =
787769575
```



```
*Jan 17 05:11:40.018: ISAKMP:(1001): processing SA payload. message ID =
787769575
*Jan 17 05:11:40.018: ISAKMP:(1001):Checking IPSec proposal 1
*Jan 17 05:11:40.018: ISAKMP: transform 1, AH_SHA
*Jan 17 05:11:40.018: ISAKMP:   attributes in transform:
*Jan 17 05:11:40.018: ISAKMP:      encaps is 1 (Tunnel)
*Jan 17 05:11:40.018: ISAKMP:      SA life type in seconds
*Jan 17 05:11:40.018: ISAKMP:      SA life duration (basic) of 900
*Jan 17 05:11:40.018: ISAKMP:      SA life type in kilobytes
*Jan 17 05:11:40.018: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50
0x0
*Jan 17 05:11:40.018: ISAKMP:      group is 5
*Jan 17 05:11:40.018: ISAKMP:      authenticator is HMAC-SHA
*Jan 17 05:11:40.018: ISAKMP:(1001):atts are acceptable.
*Jan 17 05:11:40.018: ISAKMP:(1001):Checking IPSec proposal 1
*Jan 17 05:11:40.018: ISAKMP: transform 1, ESP_AES
*Jan 17 05:11:40.018: ISAKMP:   attributes in transform:
*Jan 17 05:11:40.018: ISAKMP:      encaps is 1 (Tunnel)
*Jan 17 05:11:40.018: ISAKMP:      SA life type in seconds
*Jan 17 05:11:40.018: ISAKMP:      SA life duration (basic) of 900
*Jan 17 05:11:40.018: ISAKMP:      SA life type in kilobytes
*Jan 17 05:11:40.018: ISAKMP:      SA life duration (VPI) of  0x0 0x46 0x50
0x0
*Jan 17 05:11:40.018: ISAKMP:      authenticator is HMAC-SHA
*Jan 17 05:11:40.018: ISAKMP:      key length is 256
*Jan 17 05:11:40.018: ISAKMP:      group is 5
*Jan 17 05:11:40.018: ISAKMP:(1001):atts are acceptable.
...
```

R3 responds with a second similar message containing the same properties. R1 processes the hash and IPsec policy as shown above. R1 determines that the IPsec proposal from R3 is acceptable with the attributes in its own IPsec policies.

Notice that two transforms will both be used, with an authentication header (AH) applied to the ESP-encapsulated and encrypted data. A packet routed on R1 from 172.16.1.1 to 172.16.3.1 will first be encrypted and encapsulated as the payload of an ESP packet, and then the authentication header will be applied. The packet will be sent with a source IP address of 192.168.12.1 and destination IP address of 192.168.23.3.

```
...
*Jan 17 05:11:40.018: IPSEC(validate_proposal_request): proposal part #1
*Jan 17 05:11:40.018: IPSEC(validate_proposal_request): proposal part #1,
  (key eng. msg.) INBOUND local= 192.168.12.1, remote= 192.168.23.3,
    local_proxy= 172.16.1.0/255.255.255.0/0/0 (type=4),
    remote_proxy= 172.16.3.0/255.255.255.0/0/0 (type=4),
    protocol= AH, transform= ah-sha-hmac  (Tunnel),
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 0, flags= 0x0
*Jan 17 05:11:40.018: IPSEC(validate_proposal_request): proposal part #2
*Jan 17 05:11:40.018: IPSEC(validate_proposal_request): proposal part #2,
  (key eng. msg.) INBOUND local= 192.168.12.1, remote= 192.168.23.3,
    local_proxy= 172.16.1.0/255.255.255.0/0/0 (type=4),
    remote_proxy= 172.16.3.0/255.255.255.0/0/0 (type=4),
    protocol= ESP, transform= esp-aes 256 esp-sha-hmac  (Tunnel),
    lifedur= 0s and 0kb,
    spi= 0x0(0), conn_id= 0, keysize= 256, flags= 0x0
*Jan 17 05:11:40.022: Crypto mapdb : proxy_match
```



```
        src addr      : 172.16.1.0
        dst addr      : 172.16.3.0
        protocol      : 0
        src port      : 0
        dst port      : 0
*Jan 17 05:11:40.022: ISAKMP:(1001): processing NONCE payload. message ID =
787769575
*Jan 17 05:11:40.022: ISAKMP:(1001): processing KE payload. message ID =
787769575
*Jan 17 05:11:40.158: ISAKMP:(1001): processing ID payload. message ID =
787769575
*Jan 17 05:11:40.158: ISAKMP:(1001): processing ID payload. message ID =
787769575
...
```

R1 checks the proposal against its own IPsec policies, including the access lists on each end.

Note that the crypto map database could also evaluate the transport protocol or the source and destination ports using other properties in the extended access-list, but these were unused in the access list you supplied.

The nonce and keyed payload validate that two peers have the same keys. When the processing of this second message is complete, R1 must create the IPsec SAs so that interesting data can be encrypted using the policies negotiated.

Since there are two transforms in the transform set, two security associations and two proposals need to be created for each unidirectional path between the endpoints.

```
...
*Jan 17 05:11:40.158: ISAKMP:(1001): Creating IPSec SAs
*Jan 17 05:11:40.158:         inbound SA from 192.168.23.3 to 192.168.12.1
(f/i)  0/ 0
        (proxy 172.16.3.0 to 172.16.1.0)
*Jan 17 05:11:40.158:         has spi 0x588AA60C and conn_id 0
*Jan 17 05:11:40.158:         lifetime of 900 seconds
*Jan 17 05:11:40.158:         lifetime of 4608000 kilobytes
*Jan 17 05:11:40.158:         outbound SA from 192.168.12.1 to 192.168.23.3
(f/i) 0/0
        (proxy 172.16.1.0 to 172.16.3.0)
*Jan 17 05:11:40.162:         has spi  0x897F9209 and conn_id 0
*Jan 17 05:11:40.162:         lifetime of 900 seconds
*Jan 17 05:11:40.162:         lifetime of 4608000 kilobytes
*Jan 17 05:11:40.162: ISAKMP:(1001): Creating IPSec SAs
*Jan 17 05:11:40.162:         inbound SA from 192.168.23.3 to 192.168.12.1
(f/i)  0/ 0
        (proxy 172.16.3.0 to 172.16.1.0)
*Jan 17 05:11:40.162:         has spi 0x2E2954C0 and conn_id 0
*Jan 17 05:11:40.162:         lifetime of 900 seconds
*Jan 17 05:11:40.162:         lifetime of 4608000 kilobytes
*Jan 17 05:11:40.162:         outbound SA from 192.168.12.1 to 192.168.23.3
(f/i) 0/0
        (proxy 172.16.1.0 to 172.16.3.0)
*Jan 17 05:11:40.162:         has spi  0xAE4C8E5A and conn_id 0
*Jan 17 05:11:40.162:         lifetime of 900 seconds
*Jan 17 05:11:40.162:         lifetime of 4608000 kilobytes
```



```
*Jan 17 05:11:40.162: ISAKMP:(1001): sending packet to 192.168.23.3 my_port
500 peer_port 500 (I) QM_IDLE
*Jan 17 05:11:40.162: ISAKMP:(1001):deleting node 787769575 error FALSE reason
"No Error"
*Jan 17 05:11:40.162: ISAKMP:(1001):Node 787769575, Input =
IKE_MESG_FROM_PEER, IKE_QM_EXCH
*Jan 17 05:11:40.162: ISAKMP:(1001):Old State = IKE_QM_I_QM1  New State =
IKE_QM_PHASE2_COMPLETE
*Jan 17 05:11:40.166: IPSEC(key_engine): got a queue event with 1 KMI
message(s)
*Jan 17 05:11:40.166: Crypto mapdb : proxy_match
        src addr     : 172.16.1.0
        dst addr     : 172.16.3.0
        protocol     : 0
        src port     : 0
        dst port     : 0
*Jan 17 05:11:40.166: IPSEC(crypto_IPSec_sa_find_ident_head): reconnecting
with the same proxies and peer 192.168.23.3
*Jan 17 05:11:40.166: IPSEC(policy_db_add_ident): src 172.16.1.0, dest
172.16.3.0, dest_port 0

*Jan 17 05:11:40.166: IPSEC(create_sa): sa created,
  (sa) sa_dest= 192.168.12.1, sa_proto= 51,
    sa_spi= 0x588AA60C(1485481484),
    sa_trans= ah-sha-hmac , sa_conn_id= 2001
*Jan 17 05:11:40.166: IPSEC(create_sa): sa created,
  (sa) sa_dest= 192.168.23.3, sa_proto= 51,
    sa_spi= 0x897F9209(2306839049),
    sa_trans= ah-sha-hmac , sa_conn_id= 2002
*Jan 17 05:11:40.166: IPSEC(create_sa): sa created,
  (sa) sa_dest= 192.168.12.1, sa_proto= 50,
    sa_spi= 0x2E2954C0(774460608),
    sa_trans= esp-aes 256 esp-sha-hmac , sa_conn_id= 2001
*Jan 17 05:11:40.166: IPSEC(create_sa): sa created,
  (sa) sa_dest= 192.168.23.3, sa_proto= 50,
    sa_spi= 0xAE4C8E5A(2924252762),
    sa_trans= esp-aes 256 esp-sha-hmac , sa_conn_id= 2002
*Jan 17 05:11:40.166: IPSEC(update_current_outbound_sa): updated peer
192.168.23.3 current outbound sa to SPI AE4C8E5A
*Jan 17 05:12:30.162: ISAKMP:(1001):purging node 787769575
```

R1 has now established four security associations with R3: two in the outbound direction and two in the inbound direction. One in each direction is used for the AH transform and the other is used for the ESP transform. At this point, the ICMP replies are passing naturally through the IPsec tunnel.

Disable debugging once you are done.

```
R1# undebug all
All possible debugging has been turned off
```

As you can see, the output from the debugs is extensive and verbose. However, it can help you a lot if you are troubleshooting an IPsec problem. As mentioned earlier, we can view crypto commands and see that they now are populated with data.

```
R1# show crypto isakmp sa
dst             src             state          conn-id slot status
192.168.23.3    192.168.12.1    QM_IDLE           1001    0 ACTIVE
```



```
R1# show crypto ipsec sa

interface: FastEthernet0/0
    Crypto map tag: MYMAP, local addr 192.168.12.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.16.1.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (172.16.3.0/255.255.255.0/0/0)
   current_peer 192.168.23.3 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 4, #pkts encrypt: 4, #pkts digest: 4
    #pkts decaps: 4, #pkts decrypt: 4, #pkts verify: 4
    #pkts compressed: 0, #pkts decompressed: 0
    #pkts not compressed: 0, #pkts compr. failed: 0
    #pkts not decompressed: 0, #pkts decompress failed: 0
    #send errors 1, #recv errors 0

     local crypto endpt.: 192.168.12.1, remote crypto endpt.: 192.168.23.3
     path mtu 1500, ip mtu 1500, ip mtu idb FastEthernet0/0
     current outbound spi: 0xAE4C8E5A(2924252762)

     inbound esp sas:
      spi: 0x2E2954C0(774460608)
        transform: esp-256-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2001, flow_id: NETGX:1, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4506913/334)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE

     inbound ah sas:
      spi: 0x588AA60C(1485481484)
        transform: ah-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2001, flow_id: NETGX:1, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4506913/332)
        replay detection support: Y
        Status: ACTIVE

     inbound pcp sas:

     outbound esp sas:
      spi: 0xAE4C8E5A(2924252762)
        transform: esp-256-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2002, flow_id: NETGX:2, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4506913/332)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE

     outbound ah sas:
      spi: 0x897F9209(2306839049)
        transform: ah-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2002, flow_id: NETGX:2, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4506913/332)
        replay detection support: Y
        Status: ACTIVE

     outbound pcp sas:
```



```
R3# show crypto isakmp sa
dst             src             state          conn-id slot status
192.168.23.3    192.168.12.1    QM_IDLE              1    0 ACTIVE

R3# show crypto ipsec sa

interface: Serial0/0/1
    Crypto map tag: MYMAP, local addr 192.168.23.3

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.16.3.0/255.255.255.0/0/0)
   remote ident (addr/mask/prot/port): (172.16.1.0/255.255.255.0/0/0)
   current_peer 192.168.12.1 port 500
     PERMIT, flags={origin_is_acl,}
    #pkts encaps: 4, #pkts encrypt: 4, #pkts digest: 4
    #pkts decaps: 4, #pkts decrypt: 4, #pkts verify: 4
    #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.: 192.168.23.3, remote crypto endpt.: 192.168.12.1
     path mtu 1500, ip mtu 1500, ip mtu idb Serial0/0/1
     current outbound spi: 0x2E2954C0(774460608)

     inbound esp sas:
      spi: 0xAE4C8E5A(2924252762)
        transform: esp-256-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3001, flow_id: NETGX:1, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4385199/319)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE

     inbound ah sas:
      spi: 0x897F9209(2306839049)
        transform: ah-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3001, flow_id: NETGX:1, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4385199/318)
        replay detection support: Y
        Status: ACTIVE

     inbound pcp sas:

     outbound esp sas:
      spi: 0x2E2954C0(774460608)
        transform: esp-256-aes esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3002, flow_id: NETGX:2, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4385199/318)
        IV size: 16 bytes
        replay detection support: Y
        Status: ACTIVE

     outbound ah sas:
      spi: 0x588AA60C(1485481484)
        transform: ah-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 3002, flow_id: NETGX:2, crypto map: MYMAP
        sa timing: remaining key lifetime (k/sec): (4385199/318)
```



```
        replay detection support: Y
        Status: ACTIVE

    outbound pcp sas:
```

Why are there four security associations on each router?

Challenge: Use Wireshark to Monitor Encryption of Traffic

You can observe packets on the wire using Wireshark and see how their content looks unencrypted and then encrypted. To do this, first configure a SPAN session on the switch and open up Wireshark on a host attached to the SPAN destination port. You can use the host that you used for SDM because you don't need it anymore to configure the VPNs. If you do not know how to do this, refer to Lab 3.3: Configuring Wireshark and SPAN.

Next, you will remove the **crypto map** statements on R1 and R3. View the current configuration on the FastEthernet0/0 interface on R1 and Serial0/0/1 as shown below.

Then, issue the **no crypto map** *name* command in interface configuration mode to remove the ISAKMP security association. The router may issue a warning that ISAKMP is now off.

```
R1:

R1# show run interface fastethernet0/0
Building configuration...

Current configuration : 120 bytes
!
interface FastEthernet0/0
 ip address 192.168.12.1 255.255.255.0
 duplex auto
 speed auto
 crypto map SDM_CMAP_1
end

R1# configure terminal
R1(config)# interface fastethernet0/0
R1(config-if)# no crypto map SDM_CMAP_1
*Jan 16 06:02:58.999: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
```



```
R3:

R3# show run interface serial0/0/1
Building configuration...

Current configuration : 91 bytes
!
interface Serial0/0/1
 ip address 192.168.23.3 255.255.255.0
 crypto map SDM_CMAP_1
end

R3# configure terminal
R3(config)# interface serial0/0/1
R3(config-if)# no crypto map SDM_CMAP_1
*Jan 16 06:05:36.038: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is OFF
```

You will attempt to sniff telnet traffic from R1 to R3. Enable telnet access on R3 and configure a secure password to get to configuration mode on R3.

```
R3(config)# enable secret cisco
R3(config)# line vty 0 4
R3(config-line)# password cisco
R3(config-line)# login
```

The routers have now been configured to allow telnet access.

Have Wireshark start sniffing the packets that it receives via the SPAN session.

Choose **Capture > Interfaces...**. Then click the **Start** button associated with the interface connected to the SPAN destination port. SPAN should start capturing packets on the line, so you can now telnet from R1's loopback to R3's loopback. To source telnet traffic, use the **telnet** *destination* **/source** *interface* command.

As shown in the previous step, you must source the telnet session from R1's loopback interface to simulate the interesting traffic that will match the VPN's access list.

First, begin capturing using Wireshark. Then, begin the telnet session. Once you are connected to R3, try issuing a command or two and then logging out.

```
R1# telnet 172.16.3.1 /source Loopback0
Trying 172.16.3.1 ... Open


User Access Verification

Password: [cisco]
R3> en
Password: [cisco]

R3# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        unassigned      YES unset  administratively down down
FastEthernet0/1        unassigned      YES unset  administratively down down
Serial0/0/0            unassigned      YES unset  administratively down down
```



```
Serial0/0/1          192.168.23.3    YES manual up                    up
Serial0/1/0          unassigned      YES unset  administratively down down
Serial0/1/1          unassigned      YES unset  administratively down down
Loopback0            172.16.3.1      YES manual up                    up

R3# exit

[Connection to 172.16.3.1 closed by foreign host]
R1#
```

Now, end the capture and look at the output. You will see a set of telnet data packets. Some of these, especially the return packets, will show whole unencrypted streams of text. The reason some return packets having longer text strings is because return packets can be streamed consecutively from the router managing the connection, whereas the text you type into telnet gets sent in chunks of characters or even character by character, depending on your typing speed.

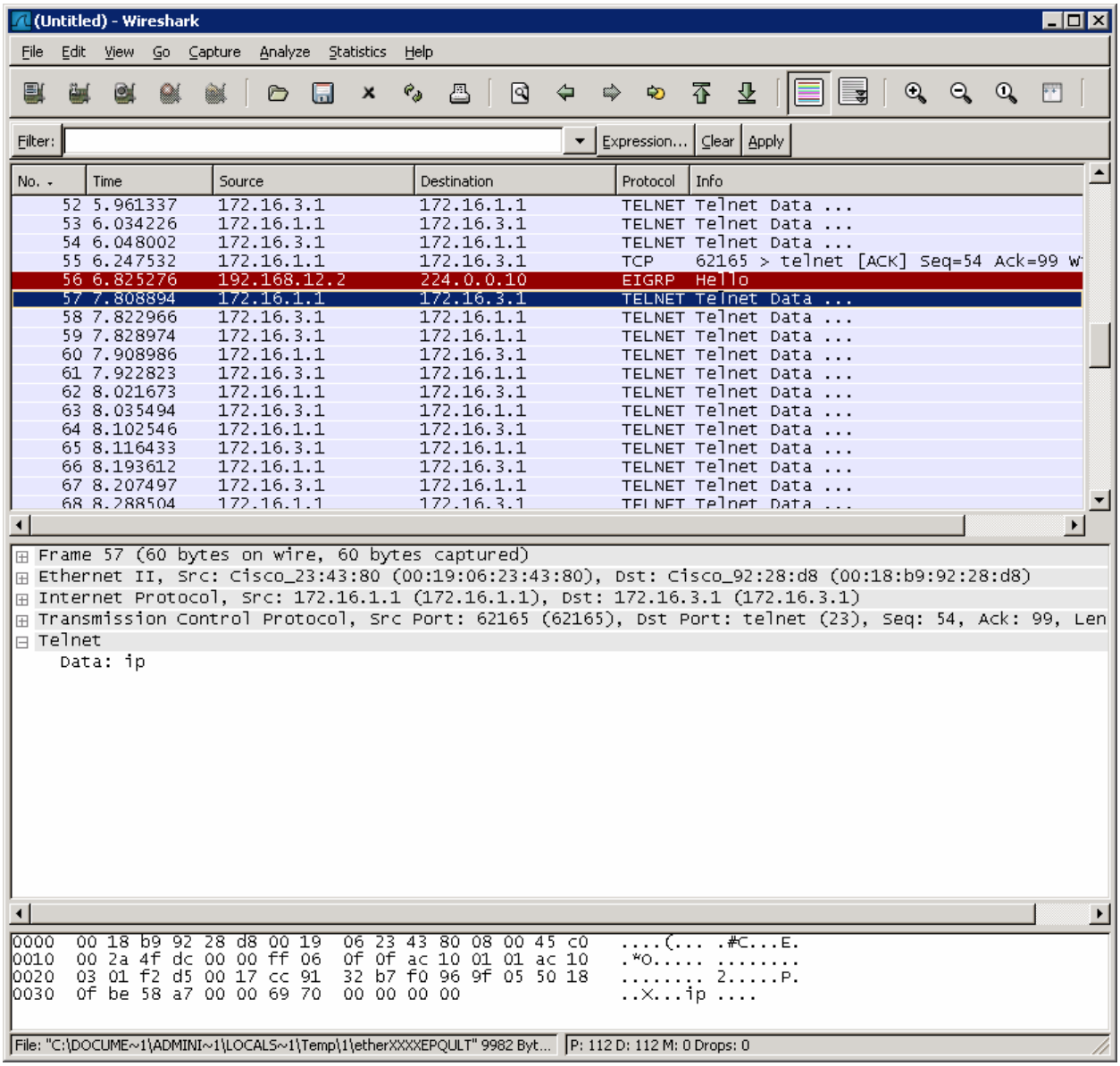


Figure 11-1: Detailed Packet Data on Telnet String Sent From R1

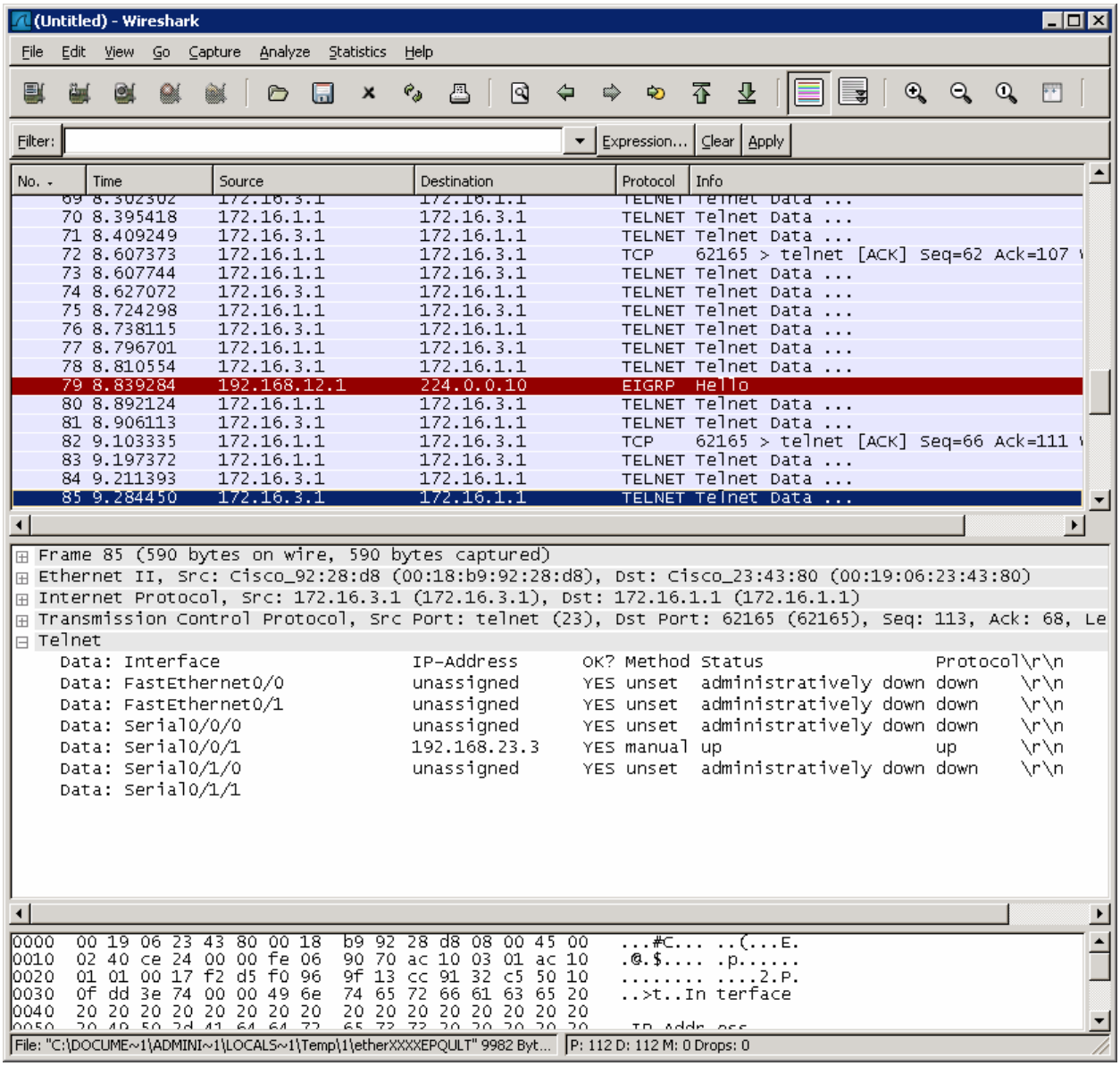


Figure 11-2: Detailed Packet Data on Return Telnet Traffic from R3

Based on this output, you can see how easy it is for someone who is in the path of sensitive data to view unencrypted or clear text traffic.

Now, you will reapply the cryptography settings on R1 and R3 and begin a telnet session from R1 to R3 as before.

Begin by reapplying the crypto maps you removed earlier on R1 and R3.

```
R1(config)# interface fastethernet0/0
R1(config-if)# crypto map SDM_CMAP_1
*Jan 16 06:36:10.295: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON

R3(config)# interface serial0/0/1
R3(config-if)# crypto map SDM_CMAP_1
```



```
*Jan 16 06:37:59.798: %CRYPTO-6-ISAKMP_ON_OFF: ISAKMP is ON
```

Start the packet capturing again in Wireshark, and then issue the same telnet sequence that you did previously.

```
R1# telnet 172.16.3.1 /source Loopback0
Trying 172.16.3.1 ... Open


User Access Verification

Password: [cisco]
R3> en
Password: [cisco]

R3# show ip interface brief
Interface              IP-Address      OK? Method Status                Protocol
FastEthernet0/0        unassigned      YES unset  administratively down down
FastEthernet0/1        unassigned      YES unset  administratively down down
Serial0/0/0            unassigned      YES unset  administratively down down
Serial0/0/1            192.168.23.3    YES manual up                    up
Serial0/1/0            unassigned      YES unset  administratively down down
Serial0/1/1            unassigned      YES unset  administratively down down
Loopback0              172.16.3.1      YES manual up                    up

R3# exit

[Connection to 172.16.3.1 closed by foreign host]
R1#
```

End your Wireshark capture when you are finished with the telnet session.

As far as the user is concerned, the telnet session seems the same with and without encryption. However, the packet capture from Wireshark shows that the VPN is actively encapsulating and encrypting packets.

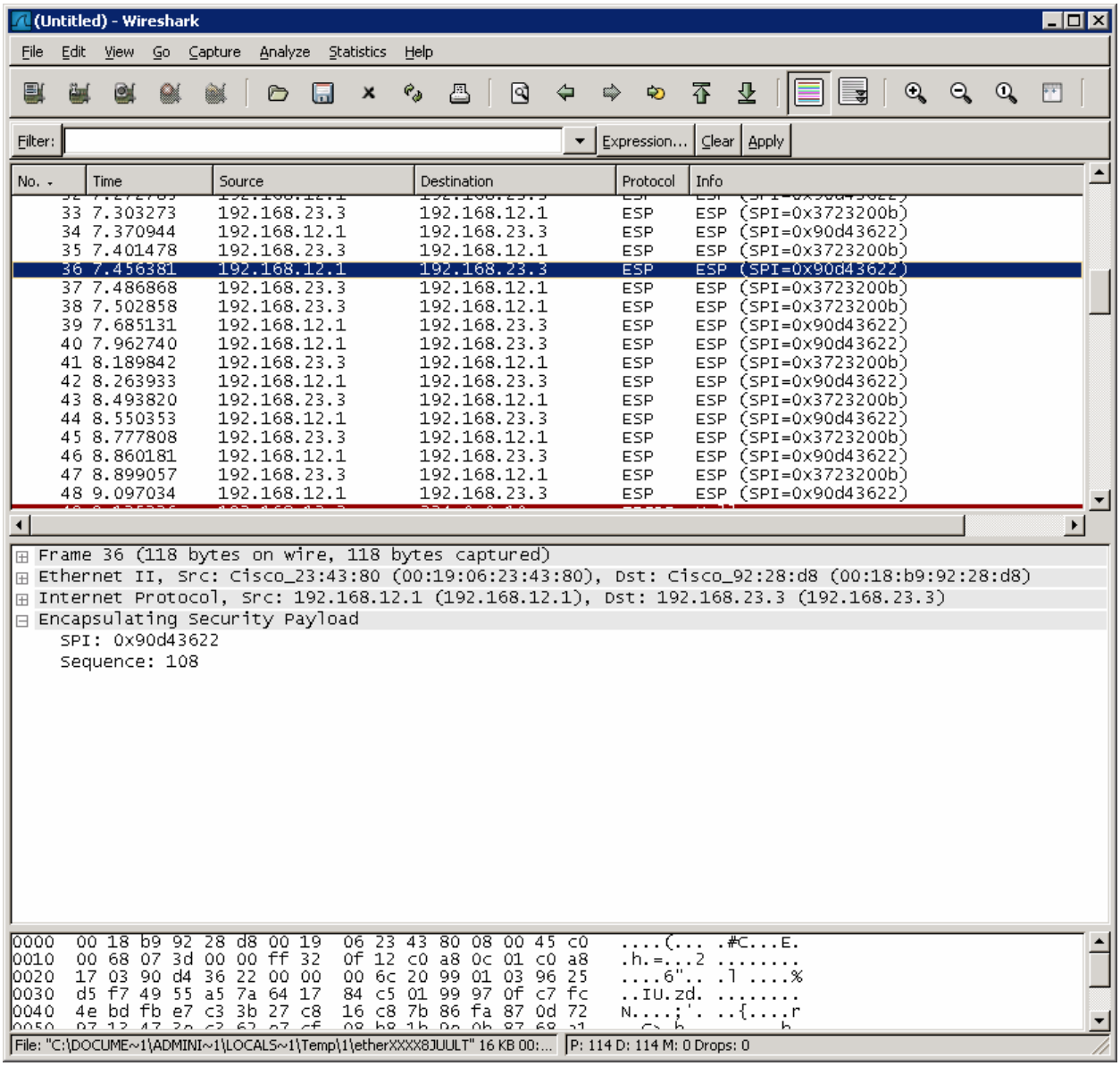


Figure 11-3: Detailed Packet Data on Encrypted Telnet String Sent From R1

Notice that the protocol is not telnet (TCP port 23), but the Encapsulating Security Protocol (ESP, IP protocol number 50). Remember, all traffic here matches the IPsec access list.

Also, notice that the source and destination are not the actual source and destination of the addresses participating in this telnet conversation. Rather, they are the endpoints of the VPN.

Why do you use the VPN endpoints as the source and destination of packets?

Finally, and most important, if you look at the contents of these packets in Wireshark, no matter how you try to format or filter them, you will not be able to see what data was originally inside.

The encryption suite provided by IPsec successfully secures data through authentication, encryption, and data-integrity services.

Appendix A: TCL Script Output

```
R1# tclsh
R1(tcl)#foreach address {
+>(tcl)#172.16.1.1
+>(tcl)#192.168.12.1
+>(tcl)#192.168.12.2
+>(tcl)#192.168.23.2
+>(tcl)#172.16.3.1
+>(tcl)#192.168.23.3
+>(tcl)#} { ping $address }

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
R1(tcl)# tclquit

R2# tclsh
R2(tcl)#foreach address {
+>(tcl)#172.16.1.1
+>(tcl)#192.168.12.1
+>(tcl)#192.168.12.2
+>(tcl)#192.168.23.2
+>(tcl)#172.16.3.1
+>(tcl)#192.168.23.3
+>(tcl)#} { ping $address }

Type escape sequence to abort.
```



```
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/68 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/28 ms
R2(tcl)# tclquit

R3# tclsh
R3(tcl)#foreach address {
+>(tcl)#172.16.1.1
+>(tcl)#192.168.12.1
+>(tcl)#192.168.12.2
+>(tcl)#192.168.23.2
+>(tcl)#172.16.3.1
+>(tcl)#192.168.23.3
+>(tcl)#} { ping $address }

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.1.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.12.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 28/28/32 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.3.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.23.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 56/58/64 ms
R3(tcl)# tclquit
```


Final Configurations:

```
R1# show run
!
hostname R1
!
crypto isakmp policy 10
 encr aes 256
 authentication pre-share
 group 5
 lifetime 3600
crypto isakmp key cisco address 192.168.23.3
!
crypto ipsec security-association lifetime seconds 1800
!
crypto ipsec transform-set 50 ah-sha-hmac esp-aes 256 esp-sha-hmac
!
crypto map MYMAP 10 ipsec-isakmp
 set peer 192.168.23.3
 set security-association lifetime seconds 900
 set transform-set 50
 set pfs group5
 match address 101
!
interface Loopback0
 ip address 172.16.1.1 255.255.255.0
!
interface FastEthernet0/0
 ip address 192.168.12.1 255.255.255.0
 crypto map MYMAP
 no shutdown
!
router eigrp 1
 network 172.16.0.0
 network 192.168.12.0
 no auto-summary
!
access-list 101 permit ip 172.16.1.0 0.0.0.255 172.16.3.0 0.0.0.255
!
end

R2# show run
!
hostname R2
!
interface FastEthernet0/0
 ip address 192.168.12.2 255.255.255.0
 no shutdown
!
interface Serial0/0/1
 ip address 192.168.23.2 255.255.255.0
 clock rate 64000
 no shutdown
!
router eigrp 1
 network 192.168.12.0
 network 192.168.23.0
 no auto-summary
!
end

R3# show run
```



```
!hostname R3
!
enable secret 5 $1$LT7i$MY2NhpGjl5uL1zNAoR2tf.
!
crypto isakmp policy 10
 encr aes 256
 authentication pre-share
 group 5
 lifetime 3600
crypto isakmp key cisco address 192.168.12.1
!
crypto ipsec security-association lifetime seconds 1800
!
crypto ipsec transform-set 50 ah-sha-hmac esp-aes 256 esp-sha-hmac
!
crypto map MYMAP 10 ipsec-isakmp
 set peer 192.168.12.1
 set security-association lifetime seconds 900
 set transform-set 50
 set pfs group5
 match address 101
!
interface Loopback0
 ip address 172.16.3.1 255.255.255.0
!
interface Serial0/0/1
 ip address 192.168.23.3 255.255.255.0
 crypto map MYMAP
 no shutdown
!
router eigrp 1
 network 172.16.0.0
 network 192.168.23.0
 no auto-summary
!
access-list 101 permit ip 172.16.3.0 0.0.0.255 172.16.1.0 0.0.0.255
!
line vty 0 4
 password cisco
 login
!
end
```

