11

Configuring a Tunnel with Generic Routing Encapsulation

Contents

Overview

The ProCurve Secure Router supports tunneling using Generic Routing Encapsulation (GRE).

GRE is a Layer 2 protocol that encapsulates higher-level protocols and renders them transparent. Routers use GRE to send traffic through an intervening network that does not support such traffic.

For example, the Internet does not route multicast messages. However, many routing protocols rely on multicasts. You can use GRE tunnels to send multicast routing updates through the Internet. You can also tunnel non-IP traffic through the Internet.

GRE Tunnels

A tunnel is a virtual point-to-point link across a multipoint-access network, such as the Internet. In a sense, a tunnel emulates a WAN link. A tunneling protocol:

- encapsulates other protocols
- sets up a point-to-point link

GRE encapsulates packets using other protocols within GRE packets. These packets, in turn, are encapsulated within IP packets. (In this way, GRE is similar to the IP Security [IPSec] protocols Authentication Header [AH] and Encapsulation Security Payload [ESP]. However, unlike AH and ESP, GRE does not ensure data integrity and confidentiality.)

GRE can encapsulate many kinds of Network Layer, Data Link Layer, or multicast protocols into an IP packet. GRE uses the same protocol identifiers that Ethernet uses. GRE then uses source routing to create a virtual point-to-point link through an IP network (such as the Internet).

On the ProCurve Secure Router, GRE encapsulates IP packets. Because GRE encapsulates packets and repackages them with a delivery IP header, it renders the original IP header transparent. GRE establishes a point-to-point link between two non-directly connected routers; these routers can then tunnel packets from hosts on private networks through the public network.

For example, on the ProCurve Secure Router, a GRE tunnel can:

- transit multicast routing protocols, such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF), through the Internet
- transit any multicast messages, such as those for a video stream
- transit traffic through a network that uses the same IP addresses (useful for integrating sites that use overlapping addresses)

GRE is often used in conjunction with IPSec. It allows two sites to tunnel routing updates and other multicasts to each other through the Internet.

You can also use GRE to establish a virtual point-to-point tunnel between two routers separated by an intervening network. For example, you can create a VPN connection through the Internet between two remote sites. You would create a tunnel whose address is on the private network, but whose source and destination endpoints are on the public network.

Advantages and Disadvantages of GRE

Because GRE can encapsulate any protocol that Ethernet can, you can use GRE to send many types traffic from one point to another. GRE connects far-flung private routers together as if the intervening hops did not exist.

However, GRE tunnels are not as secure as a tunnel established using IPSec. The GRE tunnel does not authenticate or encrypt traffic. When you use IPSec with Internet Key Exchange (IKE) to establish a VPN tunnel, IKE negotiates the keys that define that tunnel. The keys are dynamic and robust. When you use GRE, you must define the tunnel yourself, which also renders a GRE tunnel less secure than an IPSec tunnel.

Because you manually define the tunnel, GRE tunneling is less scalable that IPSec tunneling. You must create a new tunnel for every point-to-point connection that you want to establish in the VPN.

Like tunneling with IPSec, GRE can tax processing power. However, because traffic is not encrypted, GRE consumes fewer resources than IPSec.

Configuring GRE

To configure a GRE tunnel on the ProCurve Secure Router, you must:

- create a tunnel interface
- configure the tunnel source and destination endpoints
- assign the tunnel an IP address

If you want to secure the tunnel, you can also

- configure a tunnel key
- specify traffic allowed to access the tunnel
- enable checksum verification

Creating the Tunnel Interface

You establish the local endpoint for a GRE tunnel by creating a tunnel interface. From the global configuration mode context, enter:

Syntax: interface tunnel <*interface number*>

For example:

```
ProCurve(config)# interface tunnel 1
```

Configuring the Tunnel's Source and Destination and IP Address

You must identify the tunnel's source and destination. You must also configure an IP address for the tunnel interface.

The tunnel interface's IP address is distinct from the tunnel's source address. Equally, the tunnel's destination address is not the remote tunnel interface's IP address.

To understand the distinction, you must understand how GRE encapsulates packets.

When a packet arrives on the tunnel interface, GRE encapsulates it with a GRE header. This header includes a field identifying the encapsulated packet's protocol. GRE next encapsulates the GRE header with another IP header. This is the delivery header: it directs the packet through the tunnel to the remote endpoint.

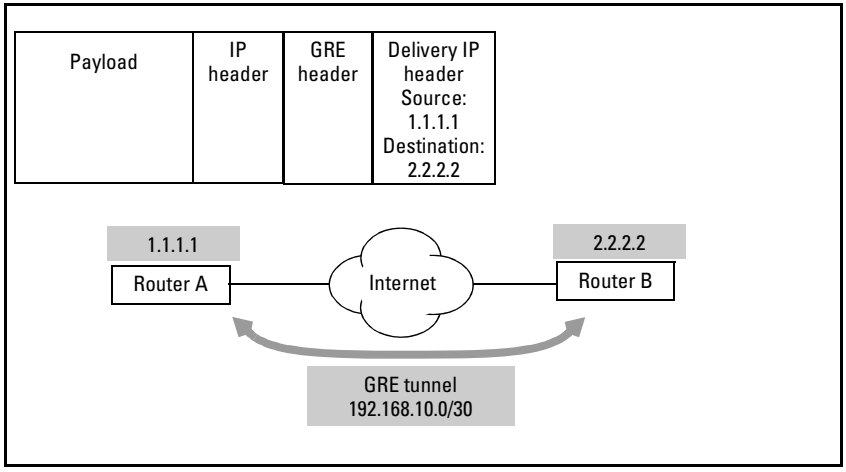

Figure 11-1. GRE Tunneling

The tunnel's source is the source address in the delivery IP header, and the tunnel's destination is the delivery IP header's destination address. (See Figure 11-1.)

The IP address for the tunnel interface is the forwarding address for packets that are to use the GRE tunnel.

Configuring the Tunnel Source

Specify the local endpoint for the tunnel. This address is the tunnel destination for the remote router's tunnel interface, and the remote router and Internet service provider (ISP) routers must be able to reach this address. Move to the tunnel interface configuration mode and enter this command to configure the tunnel source:

Syntax: tunnel source [<*A.B.C.D*> | <*interface ID*>]

The IP address that you enter is the address that the delivery IP header will include as the source address. If you enter an interface, the IP header will include the address of that interface. The interface must be configured with an IP address before you can use it as the tunnel source. The interface must be active in order for the tunnel to go up.

It is often a good idea to use a loopback interface as the tunnel endpoint. Loopback interfaces are always up, so a tunnel will not go down just because an interface on the router goes down. Use this command to create a loopback interface:

```
ProCurve(config)# interface loopback 1
```

If you are using the loopback interface as the tunnel source, you must assign it a public IP address so that ISP routers can route traffic to the tunnel endpoint. For example, configure the IP address by entering:

```
ProCurve(config-loopback 1)# ip address 1.1.1.1 /24
```

Then move to the tunnel interface configuration mode context and specify the loopback interface as the tunnel source:

```
ProCurve(config-tunnel 1)# tunnel source loopback 1
```

Configuring the Tunnel Destination

The tunnel's destination is the address that routers in the public network know how to reach. Enter:

Syntax: tunnel destination <*A.B.C.D*>

GRE will use this address for the destination address in the delivery IP header. The address must be the address configured for the remote tunnel's source. Do not enter the remote tunnel interface's address.

For example:

```
ProCurve(config-tunnel)# tunnel destination 2.2.2.2
```

Note

The router's route table must have an entry for this address. The next hop should be through a physical interface, not the address of the local tunnel endpoint. Otherwise, a recursive loop emerges as the router routes traffic back through the tunnel interface again and again.

Configuring the Tunnel's IP Address

The IP address for the tunnel interface places the tunnel in a local network. To configure the address, enter this command from the tunnel interface configuration mode context:

Syntax: ip address <*A.B.C.D*> <*subnet mask* | /*prefix length*>

The tunnel's IP address determines what traffic crosses the tunnel. For example, routing updates cross the tunnel when you enable the routing protocol on the network for the tunnel.

For the tunnel in Figure 11-1 on page 11-5, you would specify this IP address on Router A:

```
ProCurve(config-tunnel)# ip address 192.168.10.1 /30
```

Configuring the Tunnel Key

The tunnel key defines the traffic that makes up a tunnel as it passes through the intervening network. The key is inserted into a packet's GRE header and marks the packet as part of the tunnel. The key provides minimal security and ensures that the tunnel only carries the traffic it is configured to carry (particularly when more than one tunnel has been established between the same two addresses).

You configure the key from the tunnel interface configuration mode context. Enter:

Syntax: tunnel key <*key*>

For example:

```
ProCurve(config-tunnel 1)# tunnel key 1234
```

A valid key is any number from 1 to 4294967294.

Specifying Tunnel Traffic

The router encapsulates any traffic that arrives on the tunnel interface and sends it to the remote tunnel endpoint. You can select the traffic that accesses the tunnel by ensuring that the router forwards traffic through that interface.

Note

To eliminate recursive routing, the actual tunnel destination must be routed through a logical interface, *not* through the tunnel interface.

Sending Routing Updates over the Tunnel

Enable the routing protocol on the network on which the tunnel interface has its IP address.

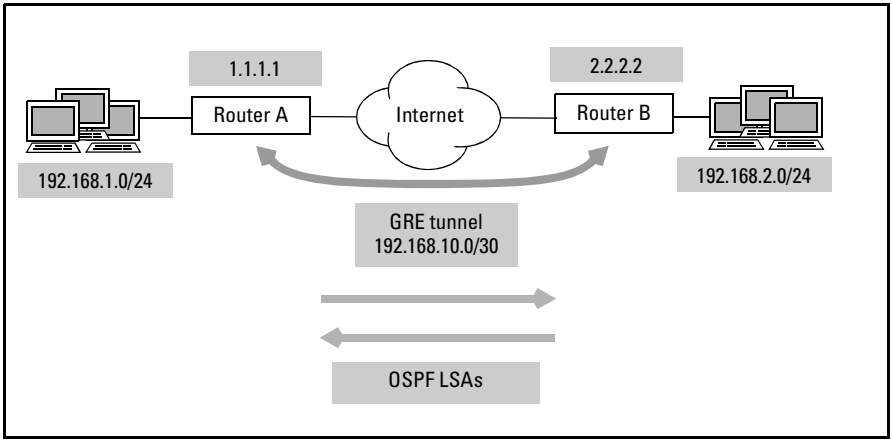

Figure 11-2. Tunneling OSPF Updates

Note that the interface's IP address is not its source. For example, in Figure 11-2 the IP address is 192.168.10.1.

To configure the router to send OSPF updates across the Internet, you would complete these steps:

1. Configure a loopback interface for the tunnel's source:

   ```
   ProCurve(config)# interface loopback 1
   ProCurve(config-loopback 1)# ip address 1.1.1.1 /30
   ```

2. Create the tunnel interface and assign it an IP address:

   ```
   ProCurve(config-loopback 1)# interface tunnel 1
   ProCurve(config-tunnel 1)# ip address 192.168.10.1 /30
   ```

3. Specify the source and destination of the tunnel:

   ```
   ProCurve(config-tunnel 1)# tunnel source loopback 1
   ProCurve(config-tunnel 1)# tunnel destination 2.2.2.2
   ```

4. Enable OSPF on the local networks, including the tunnel's network:

```
ProCurve(config)# router ospf
ProCurve(config-ospf)# network 192.168.1.0 0.0.0.255 area 0
ProCurve(config-ospf)# network 192.168.10.0 0.0.0.3 area 0
```

Sending Multicasts over the Tunnel

You can configure Protocol Independent Multicast-Sparse Mode (PIM-SM) on the tunnel interface to tunnel multicasts through the Internet. Move to the tunnel interface configuration mode context and enter this command:

```
ProCurve(config-tunnel 1)# ip pim sparse-mode
```

You should also configure the RP set, which must match the set configured on every other in the PIM domain. See *Chapter 13: Configuring Multicast Support with PIM-SM* for more information on configuring PIM-SM.

You can also configure the tunnel interface as an upstream Internet Group Management Protocol (IGMP) interface:

```
ProCurve(config-tunnel 1)# ip mcast-stub upstream
```

The tunnel interface will then join multicast host groups on behalf of connected clients and receive multicasts for these clients from the remote end of the tunnel. (Enabling PIM-SM on the interface automatically enables the interface to fulfill all functions that an IGMP upstream interface would fulfill.)

You should also enable multicast routing on the router. For example:

```
ProCurve(config)# ip multicast
```

If you are using IGMP proxy on the tunnel interface, you should configure downstream interfaces that connect to the clients and specify the remote endpoint of the tunnel as the helper address.

```
ProCurve(config)# ip mcast-stub helper-address 192.168.10.2
ProCurve(config-eth 0/1)# ip mcast-stub downstream
ProCurve(config-eth 0/1) ip mcast-stub helper-enable
```

See *Chapter 12: Configuring Multicast Support for a Stub Network* for more information on configuring multicasting.

Sending all Traffic to a Network over the Tunnel

You can add a static route to the destination network through the tunnel. From the global configuration mode context, enter:

Syntax: ip route <*destination A.B.C.D*> <*subnet mask* | /*prefix length*> tunnel <*interface number*> [<*administrative distance*>]

Specifying the administrative distance is optional.

Remember that the route to the actual tunnel destination *cannot* be through the tunnel. For example, to configure static routing for the tunnel for the network shown in Figure 11-3, you would enter these commands:

```
ProCurve(config)# ip route 192.168.2.0 /24 tunnel 1
ProCurve(config)# ip route 2.2.2.0 /24 ppp 1
```

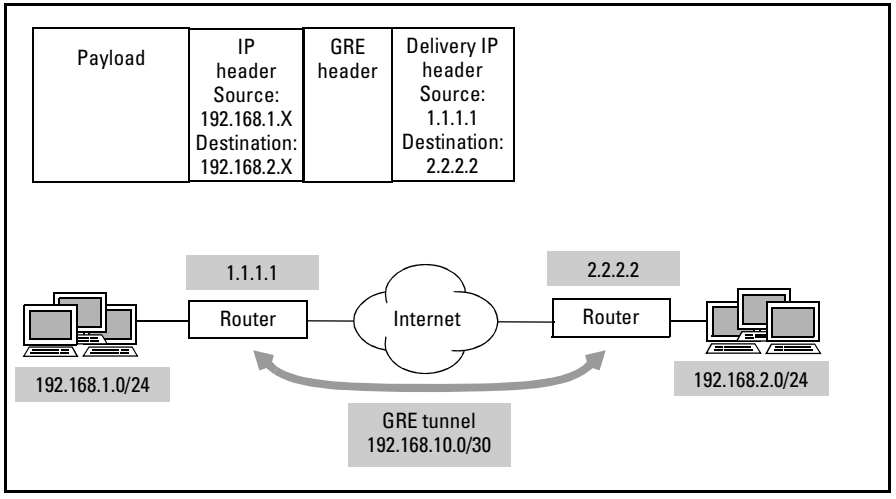

Figure 11-3. Routing Traffic over a GRE Tunnel

Filtering Traffic that Arrives on the Tunnel

You can restrict certain traffic from entering the tunnel by applying an access control policy (ACP). For example, you might want only traffic sent from a multicasting video streamer to be able to access the router through the tunnel. The multicast address for the video stream is 239.255.1.1. You should complete these steps to place the filter on the tunnel interface:

1. Configure an access control list (ACL) that selects traffic allowed over the tunnel:
   a. Name the ACL and specify whether it is standard (selects traffic by source IP address) or extended (selects traffic by a variety of IP header fields). For example:

      ```
      ProCurve(config)# ip access-list extended TunnelTraffic
      ```

   b. If necessary, add deny statements for an address or range of addresses denied the tunnel.
   c. Add permit statements for traffic allowed over the tunnel:

      Syntax: [permit | deny] ip [any | host <*source A.B.C.D*> | <*source A.B.C.D*> <*wildcard bits*>] [any | host <*destination A.B.C.D*> | <*destination A.B.C.D*> <*wildcard bits*>]

      In this example, permitted traffic is the traffic destined to the multicast address for the video stream:

      ```
      ProCurve(config-ext-nacl)# permit ip any host 239.255.1.1
      ```

2. Configure an ACP that allows this traffic:
   a. Name the ACP:

      ```
      ProCurve(config)# ip policy-class Tunnel
      ```

   b. Allow the ACL:

      Syntax: [allow | discard] list <*ACL listname*>

      For example:

      ```
      ProCurve(config-policy-class)# allow list TunnelTraffic
      ```

3. Return to the tunnel interface configuration context and apply the ACP to the tunnel:

   Syntax: interface tunnel <interface number>

   Syntax: access-policy <policyname>

Enabling Checksum Verification

A router can include a checksum in outgoing packets' GRE headers. A checksum is a value computed from the contents of a packet, and is often based on the sum of the bits. The router that receives the packet runs the same computation. If it comes up with the same value, it assumes that the data has not been altered.

Checksums help ensure data integrity. However, they are better at protecting against accidently garbled data than data that has been tampered with intentionally. Because the checksum calculation is intentionally simple, hackers can reproduce it and make sure the altered packet produces the correct checksum. For true data integrity protection, you must use a hash algorithm such as those provided with IPSec.

Checksum verification enables the router both to verify the checksum of packets it receives over a tunnel and to insert a checksum in the GRE headers of packets it transmits. To enable checksum verification, enter:

```
ProCurve(config-tunnel 1)# tunnel checksum
```

When it inserts the checksum value into the GRE header, the router also sets the checksum present bit, informing the remote endpoint that it is using checksum verification. The remote endpoint, if enabled for checksum verification, computes the packet's checksum, forwards valid packets, and drops garbled packets.

If the other end of the tunnel does not perform checksum verification, then the checksum is meaningless: all packets flow through the tunnel. Similarly, if the local endpoint has no checksum to verify, it must allow all packets.

Troubleshooting GRE Configuration

You can use the **show interfaces** command to view:

- the status of the tunnel (up or down)
- the tunnel's IP address
- packets transmitted and received over the tunnel

To track packets as the tunnel encapsulates and sends or receives and decapsulates them, use this enable mode command:

Syntax: debug interface tunnel

Note

The **show** and **debug** commands are available from enable mode; however, you can also enter them from configuration mode contexts by adding the **do** option.

The Tunnel Goes Down

If you have configured a router interface as the tunnel source, verify that this interface is up. The source interface must be up in order for the tunnel interface to be up. (Loopback interfaces are always up.) See the *Basic Management and Configuration Guide*, *Chapter 3: Configuring Ethernet Interfaces* or *Chapter 6: Configuring the Data Link Layer Protocol for E1, T1, and Serial Interfaces* for tips on troubleshooting logical interfaces.

One of the most common causes for a tunnel going down is recursive routing. When a packet arrives on the tunnel interface, the router encapsulates it and forwards it toward the tunnel destination. If the route table specifies that the tunnel interface is the next hop on the way to this destination, then the router will send the packet back through the tunnel. The packet will again be encapsulated, forwarded toward the destination, sent back to the tunnel interface, and so forth.

The ProCurve Secure Router will automatically shut down a tunnel that starts a recursive loop.

You can fix recursive routing by making sure that the forwarding interface for the tunnel destination is never a tunnel interface, but a logical interface, such as PPP 1.

The Router Does Not Receive Traffic through the Tunnel

Enter the **show interfaces** command and double-check the tunnel key. You should check the IP routing table and determine whether any traffic is being sent through the tunnel.

You can also try pinging the destination address of the tunnel to ensure that the remote endpoint is reachable. The route table should include a route to the tunnel destination through a WAN interface.

The Router Does Not Receive Routing Updates

If a router is not receiving routing updates, verify that you have enabled the routing protocol on the tunnel interface. You must enable the protocol for the network on which the tunnel has its IP address, *not* its source or destination address.

When you configure a tunnel key on a tunnel interface, the interface's maximum transmission unit (MTU) may change. This change can prevent the router from achieving full adjacency with its neighbor. You may need to reset the MTU and the connection to the neighbor so that the router can receive OSPF link state advertisements (LSAs) over the tunnel.

Refer to *Chapter 15: IP Routing—Configuring RIP, OSPF, BGP, and PBR* for general information on troubleshooting dynamic routing.

Quick Start

This section provides the commands you must enter to quickly configure a GRE tunnel and use it to carry routing updates.

Only minimal explanation is provided. If you need additional information about any of these options, check "Contents" on page 11-1 to locate the section that contains the explanation you need.

1. Configure a loopback interface:

   Syntax: interface loopback <*interface number*>

2. Assign the loopback interface an IP address:

   Syntax: ip address <*A.B.C.D*> [<*subnet mask*> /<*prefix length*>]

   For example:

   ProCurve(config-loopback)# ip address 1.1.1.1 /24

3. Create a tunnel interface:

   Syntax: interface tunnel <*interface number*>

4. Assign the tunnel interface an IP address:

   Syntax: ip address <*A.B.C.D*> [<*subnet mask*> /<*prefix length*>]

   For example:

   ProCurve(config-tunnel)# ip address 192.168.10.1 /30

5. Configure the tunnel source:

   Syntax: tunnel source [<*A.B.C.D*> | <*interface type*> <*interface number*>]

   You can use the loopback interface, which is always up:

   ProCurve(config-tunnel)# tunnel source loop 1

6. Configure the tunnel destination. The destination is the public IP address of the remote tunnel endpoint (that is, the remote tunnel interface's source, not its IP address):

   Syntax: tunnel destination <*A.B.C.D*>

7. Set the tunnel key:

   Syntax: tunnel key <*key*>

   The key is a number from 1 to 4294967294.

8. Enable the routing protocol on the network on which the tunnel has its IP address (not its source address):
   a. If you are using RIP, enter:

      ProCurve(config)# router rip

      Syntax: network <*A.B.C.D*> <*subnet mask*>

      For example:

      ProCurve(config-rip)# network 192.168.10.0 255.255.255.0

   b. If you are using OSPF, enter:

      ProCurve(config)# router ospf

      Syntax: network <*A.B.C.D*> <*subnet mask*> area <*area ID*>

      For example:

      ProCurve(config-ospf)# network 192.168.10.0 255.255.255.0 area 0

9. Make sure that the router's route table includes either a default route or a route to the tunnel's destination:

   ProCurve# show ip route

   The forwarding interface cannot be the tunnel interface. If necessary, add a route. For example:

   ProCurve(config)# ip route 2.2.2.0 /24 ppp 1