# Securing the Network in Oracle® Solaris 11.4

**Part No: E60993**
May 2021

Securing the Network in Oracle Solaris 11.4

**Part No: E60993**

# Contents

# Tables

# Examples

# Using This Documentation

- **Overview** – Describes how to provide network security. Includes link protection, tunable network parameters, firewall protection, and IPsec and IKE.
- **Audience** – Network security administrators.
- **Required knowledge** – Site security requirements.

## Product Documentation Library

Documentation and resources for this product and related products are available at `http://www.oracle.com/pls/topic/lookup?ctx=E37838-01`.

## Feedback

Provide feedback about this documentation at `http://www.oracle.com/goto/docfeedback`.

♦ ♦ ♦ CHAPTER 1

# Using Link Protection in Virtualized Environments

This chapter describes link protection and how to configure it on an Oracle Solaris system. The chapter covers the following topics:

- "What's New in Network Security in Oracle Solaris 11.4" on page 17
- "About Link Protection" on page 18
- "Configuring Link Protection" on page 19

## What's New in Network Security in Oracle Solaris 11.4

This section highlights information for existing customers about important new network security in this release.

- IKEv2 can prevent most IP layer fragmentation of its messages by replacing large encrypted messages with a series of smaller encrypted messages. These smaller IKEv2 messages can traverse network devices that drop IP fragments, such as some NAT boxes and firewalls. See "IKEv2 Service" on page 224 and Example 44, "Preventing the Loss of IKEv2 Messages From Intermediate Devices," on page 204.
- Administrators can constrain IPsec to use a particular IKE version. See "Specifying an IKE Version" on page 134 and Example 21, "Configuring IPsec Policy to Use the IKEv2 Protocol Only," on page 108.
- Packet Filter supports firewall interface groups and FTP transfers through PF doing NAT.
  - For interface groups, see "Packet Filter Macros, Tables, and Interface Groups" on page 52 and Example 11, "PF Configuration File Using Firewall Interface Groups," on page 76.
  - For FTP transfers through PF doing NAT, see "How to Make FTP Transfers Pass Through PF Doing NAT on Oracle Solaris" on page 69.

## About Link Protection

With the increasing adoption of virtualization in system configurations, guest virtual machines (VMs) can be given exclusive access to a physical or virtual link by the host administrator. This configuration improves network performance by allowing the virtual environment's network traffic to be isolated from the wider traffic that is received or sent by the host system. At the same time, this configuration can expose the system and the entire network to the risk of harmful packets that a guest environment might generate.

Link protection aims to prevent the damage that can be caused by potentially malicious guest VMs to the network. The feature offers protection from the following basic threats:

- IP, DHCP, and MAC spoofing
- L2 frame spoofing such as Bridge Protocol Data Unit (BPDU) attacks

**Note -** Link protection does not replace the deployment of a firewall, particularly for configurations with complex filtering requirements.

## Link Protection Types

The link protection mechanism in Oracle Solaris supplies the following protection types:

`mac-nospoof`

Enables protection against spoofing the system's MAC address. If the link belongs to a zone, enabling `mac-nospoof` prevents the zone's owner from modifying that link's MAC address.

`ip-nospoof`

Enables protection against IP spoofing. By default, outbound packets with DHCP addresses and link local IPv6 addresses are allowed.

You can add addresses by using the `allowed-ips` link property. For IP addresses, the packet's source address must match an address in the `allowed-ips` list. For an ARP packet, the packet's sender protocol address must be in the `allowed-ips` list.

`dhcp-nospoof`

Enables protection against spoofing of the DHCP client. By default, DHCP packets whose ID matches the system's MAC address are allowed.

You can add allowed clients by using the `allowed-dhcp-cids` link property. Entries in the `allowed-dhcp-cids` list must be formatted as specified in the `dhcpagent`(8) man page.

`restricted`

Restricts outgoing packets to IPv4, IPv6, and ARP. This protection type is designed to prevent the link from generating potentially harmful L2 control frames.

---

**Note -** Packets that are dropped because of link protection are tracked by the kernel statistics for the four protection types: `mac_spoofed`, `dhcp_spoofed`, `ip_spoofed`, and `restricted`. To retrieve these per-link statistics, see "How to View Link Protection Configuration and Statistics" on page 22.

---

For fuller descriptions of these protection types, see the `dladm`(8) man page.

## Configuring Link Protection

To use link protection, you set the `protection` property of the link. If the type of protection works with other configuration files, such as `ip-nospoof` with `allowed-ips` or `dhcp-nospoof` with `allowed-dhcp-cids`, then you perform two general actions. First, you enable link protection. Then, you customize the configuration file to identify other packets that are allowed to pass.

---

**Note -** You must configure link protection in the global zone.

---

The following task map points to the procedures for configuring link protection on an Oracle Solaris system.

**TABLE 1** Configuring Link Protection Task Map

| Task | Description | For Instructions |
|---|---|---|
| Enable link protection. | Restricts the packets that are sent from a link and protects links from spoofing. | "How to Enable Link Protection" on page 20 |
| Disable link protection. | Removes link protections. | "How to Disable Link Protection" on page 20 |
| Specify the IP link protection type. | Specifies the IP addresses that can pass through the link protection mechanism. | "How to Specify IP Addresses to Protect Against IP Spoofing" on page 21 |
| Specify the DHCP link protection type. | Specifies the DHCP addresses that can pass through the link protection mechanism. | "How to Specify DHCP Clients to Protect Against DHCP Spoofing" on page 22 |
| View the link protection configuration. | Lists the protected links and the exceptions, and shows the enforcement statistics. | "How to View Link Protection Configuration and Statistics" on page 22 |

## ▼ How to Enable Link Protection

This procedure restricts outgoing packet types and prevents the spoofing of links.

**Before You Begin** You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **View the available link protection types.**

```
# dladm show-linkprop -p protection
LINK      PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
vnic0     protection      rw   --           --           --        mac-nospoof,
                                                                   restricted,
                                                                   ip-nospoof,
                                                                   dhcp-nospoof
```

For a description of the possible types, see "Link Protection Types" on page 18 and the `dladm`(8) man page.

2. **Enable link protection by specifying one or more protection types.**

```
# dladm set-linkprop -p protection=value[,value,...] link
```

In the following example, all four link protection types on the `vnic0` link are enabled:

```
# dladm set-linkprop \
-p protection=mac-nospoof,restricted,ip-nospoof,dhcp-nospoof vnic0
```

**Caution -** Test each protection value singly before enabling it. A misconfigured system can prevent communication.

3. **Verify that the link protections are enabled.**

```
# dladm show-linkprop -p protection vnic0
LINK      PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
vnic0     protection      rw   mac-nospoof  mac-nospoof  --        mac-nospoof,
                               restricted   restricted   --        restricted,
                               ip-nospoof   ip-nospoof   --        ip-nospoof,
                               dhcp-nospoof dhcp-nospoof --        dhcp-nospoof
```

## ▼ How to Disable Link Protection

This procedure resets link protection to the default value, no link protection.

**Before You Begin** You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Disable link protection by resetting the `protection` property to its default value.**

```
# dladm reset-linkprop -p protection link
```

2. **Verify that the link protections are disabled.**

```
# dladm show-linkprop -p protection vnic0
LINK      PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
vnic0     protection      rw   --           --           --        mac-nospoof,
                                                                   restricted,
                                                                   ip-nospoof,
                                                                   dhcp-nospoof
```

## How to Specify IP Addresses to Protect Against IP Spoofing

**Before You Begin** The `ip-nospoof` protection type is enabled, as shown in "How to Enable Link Protection" on page 20.

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Verify that you have enabled protection against IP spoofing.**

```
# dladm show-linkprop -p protection link
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
link     protection      rw   ip-nospoof   ip-nospoof   --        mac-nospoof,
                                                                  restricted,
                                                                  ip-nospoof,
                                                                  dhcp-nospoof
```

2. **Add IP addresses to the list of default values for the `allowed-ips` link property.**

```
# dladm set-linkprop -p allowed-ips=IP-addr[,IP-addr,...] link
```

The following example shows how to add the IP addresses `192.0.2.11` and `192.0.2.12` to the `allowed-ips` property for the `vnic0` link:

```
# dladm set-linkprop -p allowed-ips=192.0.2.11,192.0.2.12 vnic0
```

For more information, see the `dladm`(8) man page.

## ▼ How to Specify DHCP Clients to Protect Against DHCP Spoofing

**Before You Begin** The `dhcp-nospoof` protection type is enabled, as shown in "How to Enable Link Protection" on page 20.

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Verify that you have enabled protection against DHCP spoofing.**

```
# dladm show-linkprop -p protection link
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
link     protection      rw   dhcp-nospoof dhcp-nospoof --        mac-nospoof,
                                                                  restricted,
                                                                  ip-nospoof,
                                                                  dhcp-nospoof
```

2. **Specify an ASCII phrase for the `allowed-dhcp-cids` link property.**

```
# dladm set-linkprop -p allowed-dhcp-cids=CID-or-DUID[,CID-or-DUID,...] link
```

The following example shows how to specify the string `hello` as the value for the `allowed-dhcp-cids` property for the `vnic0` link:

```
# dladm set-linkprop -p allowed-dhcp-cids=hello vnic0
```

For more information, see the `dladm`(8) man page.

## ▼ How to View Link Protection Configuration and Statistics

**Before You Begin** You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **View the link protection property values.**

```
# dladm show-linkprop -p protection,allowed-ips,allowed-dhcp-cids link
```

The following example shows the values for the `protection`, `allowed-ips`, and `allowed-dhcp-cids` properties for the `vnic0` link:

```
# dladm show-linkprop -p protection,allowed-ips,allowed-dhcp-cids vnic0
LINK    PROPERTY           PERM  VALUE         EFFECTIVE    DEFAULT POSSIBLE
vnic0   protection         rw    mac-nospoof   mac-nospoof   --     mac-nospoof,
                                 restricted    restricted           restricted,
                                 ip-nospoof    ip-nospoof           ip-nospoof,
                                 dhcp-nospoof  dhcp-nospoof         dhcp-nospoof
vnic0   allowed-ips        rw    192.0.2.11,   192.0.2.11,   --       --
                                 192.0.2.12    192.0.2.12
vnic0   allowed-dhcp-cids  rw    hello         hello         --       --
```

**Note -** The `allowed-ips` property is used only if `ip-nospoof` is enabled, as listed under `EFFECTIVE`. The `allowed-dhcp-cids` property is used only if `dhcp-nospoof` is enabled.

2. **View the link protection statistics.**

The output of the `dlstat` command is committed, so this command is suitable for scripts.

```
# dlstat -A
...
 vnic0
  mac_misc_stat
              multircv                       0
             brdcstrcv                       0
              multixmt                       0
             brdcstxmt                       0
         multircvbytes                       0
          bcstrcvbytes                       0
         multixmtbytes                       0
          bcstxmtbytes                       0
              txerrors                       0
            macspoofed                       0  <----------
             ipspoofed                       0  <----------
           dhcpspoofed                       0  <----------
            restricted                       0  <----------
              ipackets                       3
                rbytes                     182
...
```

The output indicates that no spoofed or restricted packets have attempted to pass through.

For more information, see the `dlstat(8)` man page.

♦ ♦ ♦ C H A P T E R 2

# Tuning Your Network

This chapter explains how to tune network parameters that affect security in Oracle Solaris.

## Tuning the Network

**TABLE 2** Tuning Your Network Task Map

| **Task** | **Description** | **For Instructions** |
|---|---|---|
| Disable the network routing daemon. | Limits access to systems by would-be network sniffers. | "How to Enable Dynamic Routing on a Single-Interface System" in *Configuring an Oracle Solaris 11.4 System as a Router or a Load Balancer* |
| Prevent the dissemination of information about the network topology. | Prevents the broadcast of packets. | "How to Disable Broadcast Packet Forwarding" on page 26 |
| | Prevents responses to broadcast echo requests and multicast echo requests. | "How to Disable Responses to Echo Requests" on page 26 |
| For systems that are gateways to other domains, such as a firewall or a VPN node, turn on strict source and destination multihoming. | Prevents packets that do not have the address of the gateway in their header from moving beyond the gateway. | "How to Set Strict Multihoming" on page 27 |
| Prevent DOS attacks by controlling the number of incomplete system connections. | Limits the allowable number of incomplete TCP connections for a TCP listener. | "How to Set Maximum Number of Incomplete TCP Connections" on page 28 |
| Prevent DOS attacks by controlling the number of permitted incoming connections. | Specifies the default maximum number of pending TCP connections for a TCP listener. | "How to Set Maximum Number of Pending TCP Connections" on page 28 |
| Prevent ICMP redirection. | Removes indicators of the network topology. | "How to Prevent ICMP Redirects" on page 29 |
| Return network parameters to their secure default values. | Increases security that was reduced by administrative actions. | "How to Reset Network Parameters to Secure Values" on page 30 |

| Task | Description | For Instructions |
|---|---|---|
| Limit the number of concurrent processes for system services that are controlled by `inetd`. | Limits the number of a concurrent `inetd` process. | "Modifying Services that are Controlled by inetd" in *Managing System Services in Oracle Solaris 11.4*<br><br>"Recommendations for Systems That Run inetd Based Services" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4* |

## ▼ How to Disable Broadcast Packet Forwarding

By default, Oracle Solaris forwards broadcast packets. If your site security policy requires you to reduce the possibility of broadcast flooding, change the default by using this procedure.

---

**Note -** When you disable the `_forward_directed_broadcasts` network property, you are disabling broadcast pings.

---

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the broadcast packet forwarding property to 0 for IP packets.**

```
# ipadm set-prop -p _forward_directed_broadcasts=0 ip
```

2. **Verify the current value.**

```
# ipadm show-prop -p _forward_directed_broadcasts ip
PROTO  PROPERTY                     PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ip    _forward_directed_broadcasts  rw   0         --           0         0,1
```

**See Also** `ipadm(8)` man page

## ▼ How to Disable Responses to Echo Requests

Use this procedure to prevent the dissemination of information about the network topology.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the response to broadcast echo requests property to 0 for IP packets, then verify the current value.**

```
# ipadm set-prop -p _respond_to_echo_broadcast=0 ip

# ipadm show-prop -p _respond_to_echo_broadcast ip
PROTO  PROPERTY                  PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ip    _respond_to_echo_broadcast rw   0         --           1         0,1
```

2. **Set the response to multicast echo requests property to 0 for IP packets, then verify the current value.**

```
# ipadm set-prop -p _respond_to_echo_multicast=0 ipv4
# ipadm set-prop -p _respond_to_echo_multicast=0 ipv6

# ipadm show-prop -p _respond_to_echo_multicast ipv4
PROTO  PROPERTY                  PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4  _respond_to_echo_multicast rw   0         --           1         0,1
# ipadm show-prop -p _respond_to_echo_multicast ipv6
PROTO  PROPERTY                  PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv6  _respond_to_echo_multicast rw   0         --           1         0,1
```

**See Also** For more information, see "_respond_to_echo_broadcast (IP) and _respond_to_echo_multicast Parameters (IPv4 or IPv6)" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual* and the `ipadm(8)` man page.

## ▼ How to Set Strict Multihoming

For systems that are gateways to other domains, such as a firewall or a VPN node, use this procedure to turn on strict multihoming. The `hostmodel` property controls the send and receive behavior for IP packets on a multihomed system.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the `hostmodel` property to `strong` for IP packets.**

```
# ipadm set-prop -p hostmodel=strong ipv4
# ipadm set-prop -p hostmodel=strong ipv6
```

2. **Verify the current value and note the possible values.**

```
# ipadm show-prop -p hostmodel ip
```

```
PROTO  PROPERTY     PERM CURRENT    PERSISTENT   DEFAULT    POSSIBLE
ipv6   hostmodel    rw   strong     strong       weak       strong,src-priority,weak
ipv4   hostmodel    rw   strong     strong       weak       strong,src-priority,weak
```

**See Also** For more information, see "hostmodel Parameter (IPv4 or IPv6)" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual* and the `ipadm(8)` man page.

For more information about the use of strict multihoming, see "How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode" on page 116.

## How to Set Maximum Number of Incomplete TCP Connections

Use this procedure to prevent denial of service (DOS) attacks by controlling the number of pending connections that are incomplete.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the maximum number of incoming connections.**

```
# ipadm set-prop -p _conn_req_max_q0=4096 tcp
```

2. **Verify the current value.**

```
# ipadm show-prop -p _conn_req_max_q0 tcp
PROTO  PROPERTY         PERM CURRENT    PERSISTENT   DEFAULT    POSSIBLE
tcp   _conn_req_max_q0  rw   4096       --           128        1-4294967295
```

**See Also** For more information, see "_conn_req_max_q0 Parameter" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual* and the `ipadm(8)` man page.

## How to Set Maximum Number of Pending TCP Connections

Use this procedure to prevent DOS attacks by controlling the number of permitted incoming connections.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the maximum number of incoming connections.**

```
# ipadm set-prop -p _conn_req_max_q=1024 tcp
```

2. **Verify the current value.**

```
# ipadm show-prop -p _conn_req_max_q tcp
PROTO  PROPERTY         PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
tcp   _conn_req_max_q   rw   1024      --           128       1-4294967295
```

**See Also** For more information, see "_conn_req_max_q Parameter" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual* and the ipadm(8) man page.

## ▼ How to Prevent ICMP Redirects

Routers use ICMP redirect messages to inform hosts of more direct routes to a destination. An illicit ICMP redirect message could result in a man-in-the-middle attack.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the ignore redirects property to 1 for IP packets, then verify the current value.**

   ICMP redirect messages modify the host's route table and are unauthenticated. Additionally, the processing of redirected packets increases CPU demands on systems.

```
# ipadm set-prop -p _ignore_redirect=1 ipv4
# ipadm set-prop -p _ignore_redirect=1 ipv6
# ipadm show-prop -p _ignore_redirect ipv4
PROTO  PROPERTY         PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4  _ignore_redirect  rw   1         1            0         0,1
# ipadm show-prop -p _ignore_redirect ipv6
PROTO  PROPERTY         PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv6  _ignore_redirect  rw   1         1            0         0,1
```

2. **Prevent sending ICMP redirect messages.**

   These messages include information from the route table that could reveal part of the network topology.

```
# ipadm set-prop -p send-redirects=off ipv4
# ipadm set-prop -p send-redirects=off ipv6
# ipadm show-prop -p send-redirects ipv4
PROTO PROPERTY          PERM CURRENT  PERSISTENT   DEFAULT  POSSIBLE
ipv4  send-redirects    rw   off      off          on       on,off

# ipadm show-prop -p send-redirects ipv6
PROTO  PROPERTY         PERM CURRENT  PERSISTENT   DEFAULT  POSSIBLE
ipv6  send-redirects   rw   off       off          on       on,off
```

For more information, see "send-redirects Parameter (IPv4 or IPv6)" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual* and the `ipadm(8)` man page.

## How to Reset Network Parameters to Secure Values

Many network parameters that are secure by default are tunable, and might have been changed from the default. If site conditions permit, return the following tunable parameters to their default values.

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the source packet forwarding property to 0 for IP packets, then verify the current value.**

   The default value prevents DOS attacks from spoofed packets.

```
# ipadm set-prop -p _forward_src_routed=0 ipv4
# ipadm set-prop -p _forward_src_routed=0 ipv6
# ipadm show-prop -p _forward_src_routed ipv4
PROTO  PROPERTY             PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4  _forward_src_routed   rw   0         --           0         0,1
# ipadm show-prop -p _forward_src_routed ipv6
PROTO  PROPERTY             PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv6  _forward_src_routed   rw   0         --           0         0,1
```

   For more information, see "_forwarding_src_routed Parameter (IPv4 or IPv6)" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual*.

2. **Set the netmask response property to 0 for IP packets, then verify the current value.**

The default value prevents the dissemination of information about the network topology.

```
# ipadm set-prop -p _respond_to_address_mask_broadcast=0 ip
# ipadm show-prop -p _respond_to_address_mask_broadcast ip
PROTO PROPERTY                          PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ip    _respond_to_address_mask_broadcast rw   0         --           0         0,1
```

3. **Set the timestamp response property to 0 for IP packets, then verify the current value.**

   The default value removes additional CPU demands on systems and prevents the dissemination of information about the network.

```
# ipadm set-prop -p _respond_to_timestamp=0 ip
# ipadm show-prop -p _respond_to_timestamp ip
PROTO  PROPERTY                       PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ip    _respond_to_timestamp           rw   0         --           0         0,1
```

4. **Set the broadcast timestamp response property to 0 for IP packets, then verify the current value.**

   The default value removes additional CPU demands on systems and prevents dissemination of information about the network.

```
# ipadm set-prop -p _respond_to_timestamp_broadcast=0 ip
# ipadm show-prop -p _respond_to_timestamp_broadcast ip
PROTO  PROPERTY                       PERM CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ip    _respond_to_timestamp_broadcast  rw   0         --           0         0,1
```

5. **Prevent IP source routing.**

   The default value prevents packets from bypassing network security measures. Source-routed packets allow the source of the packet to suggest a path different from the path configured on the router.

   ---

   **Note -** This parameter might be set to `1` for diagnostic purposes. After diagnosis is complete, return the value to `0`.

   ---

```
# ipadm set-prop -p _rev_src_routes=0 tcp
# ipadm show-prop -p _rev_src_routes tcp
PROTO PROPERTY         PERM CURRENT  PERSISTENT  DEFAULT  POSSIBLE
tcp   _rev_src_routes  rw   0        --          0        0,1
```

   For more information, see "_rev_src_routes Parameter" in *Oracle Solaris 11.4 Tunable Parameters Reference Manual*.

6. **Set the value of `TCP_STRONG_ISS` to `2`, then reboot the system.**

The default value, which is set in the `/etc/default/inetinit` file, ensures that the TCP initial sequence number generation parameter complies with Defending against Sequence Number Attacks (`https://www.rfc-editor.org/info/rfc6528`).

---

**Note -** You must become an administrator who is assigned the `solaris.admin.edit/etc.default/inetinit` authorization. By default, the `root` role has this authorization. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

---

```
# pfedit /etc/default/inetinit
TCP_STRONG_ISS=2
# /usr/sbin/reboot
```

**See Also** `ipadm`(8) man page

♦ ♦ ♦ C H A P T E R 3

# Protecting Networks With IEEE 802.1X Certificates

You can require clients to provide authentication to access a network. Access is authenticated by an IEEE 802.1X certificate, which you configure and install on the datalink that provides access to the network. Only clients with the appropriate certificate can access the network.

## Administering Port-Based Authentication on Datalinks

The IEEE 802.1X feature restricts the use of IEEE 802 LAN service access points and secures communications between authenticated devices. In this release, support for port-based authentication is client-side only and limited to wired datalinks.

A typical setup includes the following components:

- Client system that requests access to a secured network
- Network access point such as a switch that sends authentication requests and responses between the client and the authentication server
- Authentication server that runs an authentication, authorization, and accounting (AAA) protocol such as Radius to authenticate the client

The credential information that is required for port-based authentication is grouped by network. Each network is identified by a network name.

To configure the credentials, you use the `nacadm` command. See the `nacadm`(8) man page.

After configuring credential information, you set the datalink's `authentication` property, which enables port-based authentication on that datalink.

You must configure at least one network to use IEEE 802.1X port-based authentication. If you configure multiple networks to use port-based authentication, each network must have the required credentials for the system to be authenticated. See "How to Configure and Enable IEEE 802.1X Port-Based Authentication" on page 34.

The IEEE 802.1X port-based authentication feature is managed through the `network/network-access-control:default` SMF service. This service is disabled until you install the `network-access-control` software package on the client system. The service is also automatically enabled whenever you enable authentication on a specific datalink.

The IEEE 802.1X port-based authentication process starts when the `nacd` daemon is running on the client system. If the authentication is successful, the system receives DHCP service from the DHCP server. If the authentication fails, the system boots with no network. The behavior resembles the case where DHCP fails and then times out. The `nacd` daemon logs authentication failures in the `rsyslog` file for easy tracking. See `syslog`(3C).

If your network is configured to use DHCP, and the DHCP server resides on a secured local area network (LAN) on which the system attempts to connect, the system must first authenticate through IEEE 802.1X and then connect to the secure LAN. From the LAN, the system can communicate with the DHCP server.

On link aggregations, if port authentication is enabled but fails, the link is treated as though the port is disabled and no traffic passes through the port.

Before performing DR on a system, you must reset the datalink's `authentication` property. After DR has completed, set the `authentication` property to re-enable IEEE 802.1X authentication on the datalink. See Example 1, "Disabling IEEE 802.1X Port-Based Authentication on a Datalink," on page 36 and the `dladm`(8) man page.

## ▼ How to Configure and Enable IEEE 802.1X Port-Based Authentication

**Before You Begin** You must be an administrator with the rights to install packages, start services, and administer the network. The `root` role has all of these rights. The System Administration, Service Configuration, and Network Management rights profiles provide these rights. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Perform the required IEEE 802.1X setup tasks, which typically include the following:

- Designating a host system to be used as the client (or supplicant) that requests access to a secured network.
- Determining a network access point (a switch) to be used as the authenticator that passes the authentication requests and responses between the client and the authentication server.
- Designating an authentication server that is running an AAA application such as Radius to authenticate the client.

- Determining which method of network security the system that you are connecting to uses, for example, MD5 or TLS, and the corresponding credentials (ID and password) that are required.

1. **On the client system, install the `network-access-control` software package.**

```
client$ pkg install network-access-control
```

2. **(Optional) Verify that the appropriate service is running.**

```
client$ svcs network-access-control
STATE  STIME FMRI
online Sep_07 svc:/network/network-access-control:default
```

**Note -** To enable the service, type:

```
client$ svcadm enable network-access-control
```

3. **On the client system, add a network to the specified datalink and configure its credentials.**

```
client$ nacadm add-net -p key-mgmt=DOT1X,eap=tls,\
   ca-cert=file-location,client-cert=cert-location,\
   private-key-file=key-location,private-key-passwd=key-file  network-name
```

| | |
|---|---|
| `key-mgmt=DOT1X` | Specifies an acceptable authenticated key management protocol. Currently, `DOT1X` is the only key management protocol that is supported. |
| `eap=tls` | Space-separated list of acceptable Extensible Authentication Protocol (EAP) methods. Use `tls` for Transport Layer Security (TLS). |
| *file-location* | Location of the TLS certificate authority (CA) certificate. The default location is `/etc/certs/ca-certificates.crt`. |
| *cert-location* | Location of the client certificate. |
| *key-location* | Location of the client private key. |
| *key-file* | File containing the client private key password. |
| *network-name* | Name that you assign to the given network. |

**Note -** Do not use `auto` or `automatic` for network names. These names are reserved.

For example, associate the certificate with the network name `netsec0`:

```
$ nacadm add-net -p key-mgmt=DOT1X,eap=tls,client-cert=/etc/certs/localhost/nacd0.pem,\
   private-key-file=/etc/nacd/priv-key.pem,private-key-passwd=.admin/dot1x.pwd netsec0
```

4. **Set the `authentication` property to the network name that you added in Step 3.**

```
$ dladm set-linkprop -p authentication=netsec0 net0
```

This step enables the IEEE 802.1X process on the `net0` datalink.

5. **Verify the `authentication-state` of the datalink.**

```
$ dladm show-linkprop -p authentication-state
LINK  AUTHENTICATION-STATE
net0  succeeded
net1  off
net2  off
```

Possible values for the `AUTHENTICATION-STATE` field are `succeeded` (default), `off`, `failed`, and `in-process`.

**Example 1** Disabling IEEE 802.1X Port-Based Authentication on a Datalink

You can disable IEEE 802.1X port-based authentication on a datalink in two ways.

- Switching the `authentication` property off.

  ```
  $ dladm set-linkprop -p authentication=off net0
  ```

- Resetting the `authentication` property.

  ```
  $ dladm reset-linkprop -p authentication net0
  ```

**Example 2** Displaying Configuration Information for Networks That Use IEEE 802.1X Port-Based Authentication

This example shows how to display configuration information for a network that uses IEEE 802.1X port-based authentication.

```
$ nacadm show-net netsec0
NAME    KEY-MGMT EAP IDENTITY CA-CERT                                          CERT
                      PRIV-KEY
netsec0 dot1x    tls id       /etc/certs/ca-certificates/CA/Swissign_Gold1.pem /etc/
certs/localhost/nacd0.pem /etc/nacd/priv-key.pem
```

♦ ♦ ♦ C H A P T E R 4

# Oracle Solaris Firewall

This chapter provides an overview of the Packet Filter (PF) feature of Oracle Solaris. For PF tasks, see Chapter 5, “Configuring the Firewall in Oracle Solaris”.

This chapter covers the following topics:

## Introduction to Packet Filter

The OpenBSD Packet Filter (PF) feature of Oracle Solaris is a network firewall that captures incoming packets and evaluates them for entry to and exit from the system. PF provides stateful packet inspection. It can match packets by IP address and port number as well as by the receiving network interface.

Oracle Solaris PF is based on OpenBSD Packet Filter (PF) version 5.5, which is enhanced to work with Oracle Solaris components, such as zones with exclusive IP instances.

At installation, Oracle Solaris PF behaves differently from OpenBSD PF at installation.

- In Oracle Solaris, the `svc:/network/firewall` service is installed but disabled by default.
- If you enable the service with the default configuration that Oracle Solaris ships, then the `firewall` service is put in the `degraded` state, as described in “Packet Filter Configuration File and the `firewall` Service” on page 46.

If you are transferring IP Filter firewall rules to PF firewall rules, you must ensure that the PF rules enforce the same policy. Identical rules in IP Filter and PF can enforce different policy.

The following OpenBSD PF features are not included in the Oracle Solaris version:

- Network address translation (NAT-64) between IPv6 and IPv4 as described by RFC 6146
- PFSYNC, which allows PF firewalls to be deployed as a cluster
- QOS (packet queuing)
- Netflow statistics

## Comparing PF in Oracle Solaris to IP Filter and to OpenBSD Packet Filter

The earlier releases of Oracle Solaris used IP Filter as the firewall. In this release, PF is the only supported firewall.

### Comparing IP Filter and Oracle Solaris Packet Filter

If you plan to transfer IP Filter rules to Packet Filter (PF) rules, note that the features of IP Filter and PF do not match exactly. Therefore, no reliable conversion tool to map IP Filter configurations to PF configurations is possible. The best strategy when converting network policies, including firewall policies, from one product to another is to review the requirements and specifications and then implement policies using the new tool.

The following table compares the Oracle Solaris implementation of PF with IP Filter. Table 4, "Differences Between OpenBSD PF and Oracle Solaris PF," on page 39 compares the Oracle Solaris implementation of PF with OpenBSD PF.

**TABLE 3** Comparison of IP Filter and Packet Filter on Oracle Solaris

| Firewall Feature | IP Filter | Oracle Solaris PF Implementation |
|---|---|---|
| Configuration files | Several, such as `ippool.conf`, `ipnat.conf`, and `ipv6.conf` | One `pf.conf` file |
| Ease of understanding the rules | Complex syntax | Shortcuts such as macros and tables aid readability |
| IPv4 and IPv6 packet fragments | Administrator must explicitly turn on reassembly | IP reassembly is on by default |

| Firewall Feature | IP Filter | Oracle Solaris PF Implementation |
|---|---|---|
| Loopback interface protection | Must be enabled by `set intercept_loopback true;` | Firewall always intercepts packets on loopback interface |
| Package name | `ipfilter` | `firewall` |
| OS signature file | None | `pf.os` |
| `pass` rules | Stateless by default | Stateful by default |
| Rights profile | IP Filter Management | Network Firewall Management |
| SMF service name | `ipfilter` | `firewall`, which is put in the `degraded` state when enabled with the default configuration that Oracle Solaris ships. |
| FTP packet filtering over NAT | Handled in the kernel | Handled by an `ftp-proxy` daemon |
| Packet logging | Uses `/dev/ipl` character device to pass logged packets to `ipmon` service. | Uses capture links (pseudo links) to pass packets from kernel to userland. Packets are then read by `pflog` service or can be read by `tcpdump` and Wireshark. |

## Comparing Oracle Solaris Packet Filter and OpenBSD Packet Filter

The following table describes the differences between the OpenBSD implementation of PF and the Oracle Solaris version. For OpenBSD features that Oracle Solaris does not include, see "Introduction to Packet Filter" on page 37.

**TABLE 4** Differences Between OpenBSD PF and Oracle Solaris PF

| OpenBSD PF Behavior | Oracle Solaris PF Behavior | Difference in Oracle Solaris PF |
|---|---|---|
| OpenBSD provides PF as part of a base system. | PF is installed as an IPS package. | IPS repositories provide security for data at rest and data in transit. |
| `pfctl` command manages the firewall. | `svc*` commands manage the firewall, which is an SMF service. | SMF commands supplement `pfctl` functionality. |
| NAT works over IPv4 and IPv6 networks. | OpenBSD supports NAT-64 as described by RFC 6146, while Oracle Solaris supports traditional NAT only, as described by RFC 2663. | PF supports NAT on IPv4 networks only. |
| Firewall interface groups are managed by `ifconfig`. | Firewall interface groups are managed by `ipadm`. | No differences except the command to use. |
| No provision for zones. | PF works in and between Oracle Solaris Zones. | Non-global zones can use PF.<br><br>Filtering between zones is supported in zones that function as virtual routers for the other zones on the system. |

For additional information, see "Guidelines for Using Packet Filter in Oracle Solaris" on page 40 and Chapter 5, "Configuring the Firewall in Oracle Solaris".

## Guidelines for Using Packet Filter in Oracle Solaris

When using PF, note the following guidelines:

- To enable and use the PF firewall, see "How to Configure the Firewall on Oracle Solaris" on page 68.

  PF is installed but disabled. The `solaris-small-server`, `solaris-large-server`, and `solaris-desktop` group packages install the PF firewall by default.
- Use SMF commands, such as `svcadm enable firewall`, to manage PF. For when to use the `pfctl` command, see "Using PF Features to Administer the Firewall" on page 61.

  For an overview of SMF, see Chapter 1, "Introduction to the Service Management Facility" in *Managing System Services in Oracle Solaris 11.4*. For SMF procedures, see Chapter 3, "Administering Services" in *Managing System Services in Oracle Solaris 11.4*.
- To administer PF, become an administrator who is assigned the Network Firewall Management rights profile. The `root` role includes this profile.

  Best practice is to assign the Network Firewall Management rights profile to a user or to a role that you create. To create the role and assign the role to a user, see "Creating a Role" in *Securing Users and Processes in Oracle Solaris 11.4*.
- To edit the `pf.conf` configuration file, use the `pfedit` command. After editing, use the `pfctl -nf` command to verify the syntax and refresh the `firewall` service.
- Use macros, tables, and firewall interface groups to simplify rules and enhance performance. For more information, see "Packet Filter Macros, Tables, and Interface Groups" on page 52.
- Use firewall interface groups in PF to specify the same firewall policy on multiple hosts that handle similar traffic on interfaces with different names. The group names are used in policy rules. Without the group names, these same rules could not apply to multiple hosts when the interface names differ.

  For more information, see "Packet Filter Macros, Tables, and Interface Groups" on page 52. For tasks, see "How to Use Groups to Simplify Firewall Policy in a Network" on page 73 and Example 11, "PF Configuration File Using Firewall Interface Groups," on page 76.

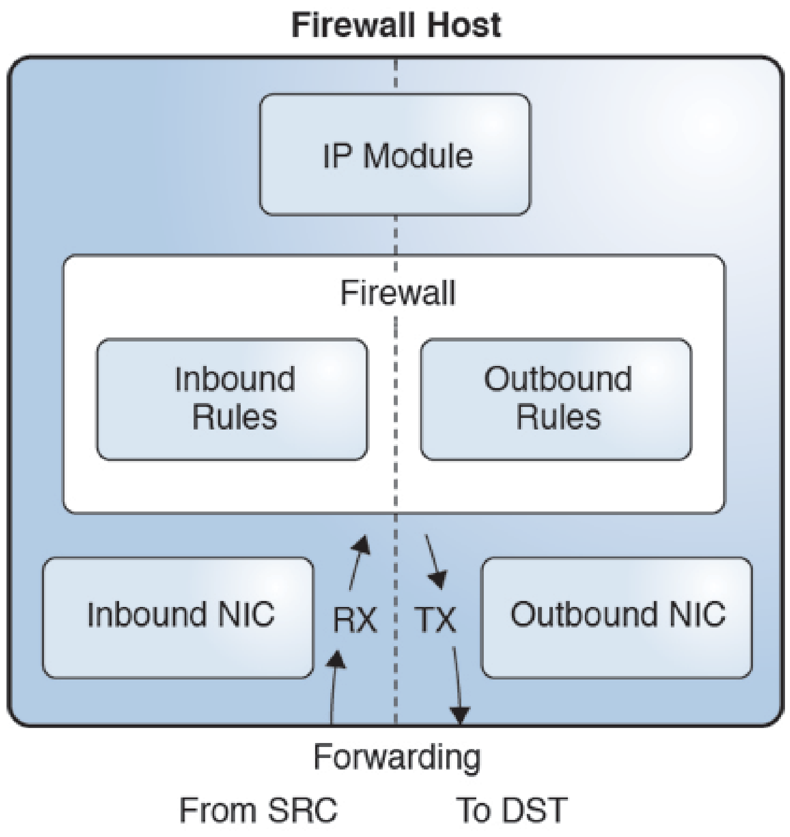
## Packet Filter Firewall and Packet Processing

This section illustrates how packets arrive on a firewall host and are processed by the firewall, which sits between network devices and the IP module. Figure 1, "OpenBSD Packet Firewall," on page 42 illustrates how the firewall module can inspect all packets which travel between network devices and the host's IP stack. Figure 2, "Packet Flow in the OpenBSD Packet Firewall," on page 44 illustrates how packets are processed inside the firewall module.

### Packet Filter Firewall Module in Oracle Solaris

The following illustration shows how a firewall host forwards a packet from a remote source to a remote destination. Forwarding is shown at the bottom of the graphic.

**FIGURE 1** OpenBSD Packet Firewall

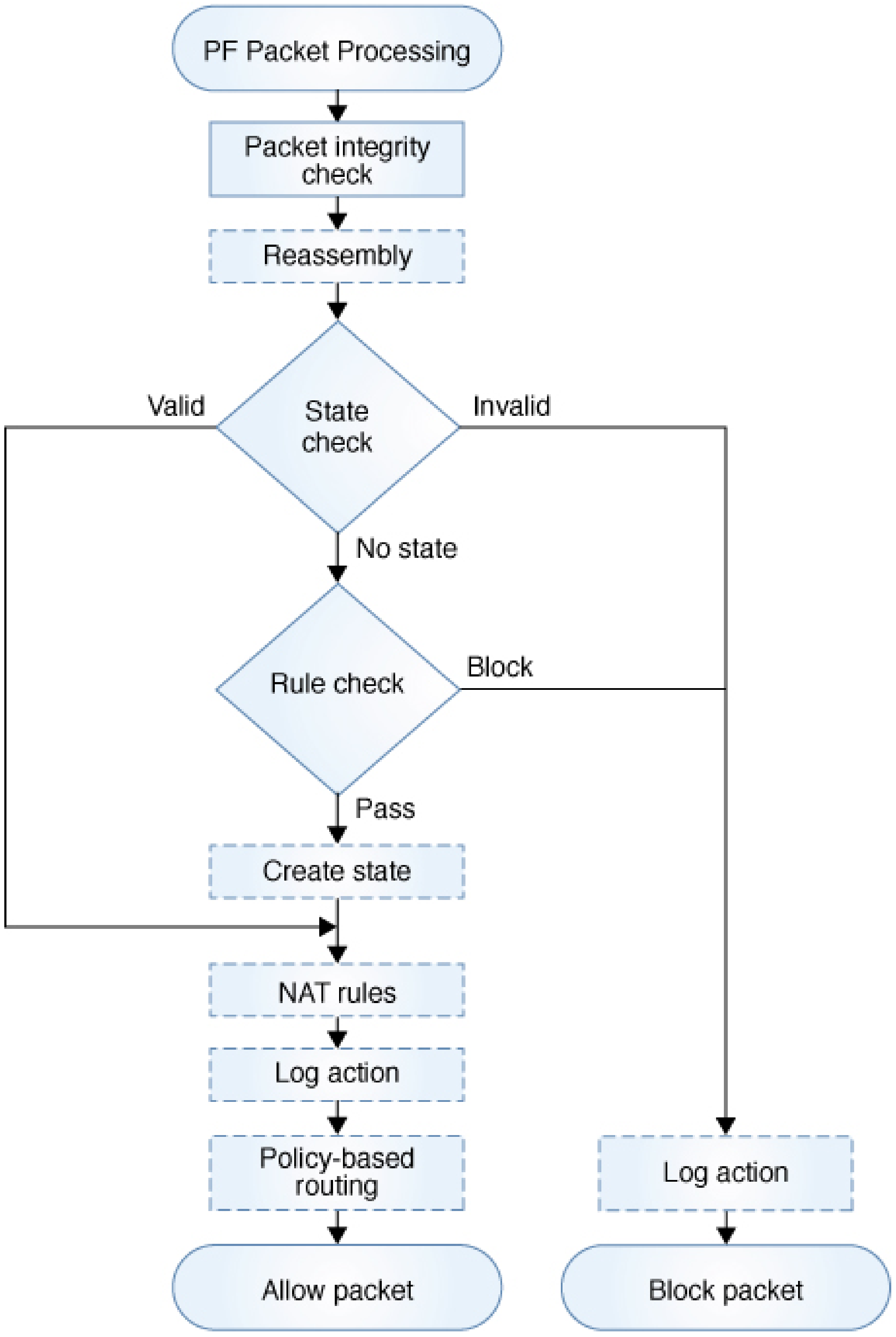


The left side is the receiving side (RX), the inbound path. The right side is the transmitting side (TX). The packet enters the host from the inbound NIC. As the inbound packet moves toward the IP module, the firewall intercepts it and applies inbound rules. If the inbound policy accepts the packet, the firewall sends the inbound packet to the IP module.

The IP module uses the destination address in the IP header to find a route. If the IP module finds a matching route, it transmits the packet (TX). As the packet travels towards the outbound NIC, the firewall intercepts it again. This time the firewall applies outbound rules to the packet. If the outbound policy accepts the packet, the firewall sends it to the outbound NIC, which transmits the packet to the remote destination.

## Packet Processing in PF

Figure 2, “Packet Flow in the OpenBSD Packet Firewall,” on page 44 illustrates the packet inspection process by the PF firewall module. The dashed boxes indicate administrative options. PF rules can include options to reassemble IP packet fragments, process NAT rules, log actions, and create a state.


PF Packet Processing
Packet integrity check
Reassembly
State check
Valid
Invalid
No state
Rule check
Block
Pass
Create state
NAT rules
Log action
Policy-based routing
Allow packet
Log action
Block packet

As soon as a packet enters the firewall, the firewall runs a basic packet integrity check. It also reassembles fragments into complete packets if the administrator has specified the `set reassemble yes` directive. These integrity checks discard packets with invalid flags settings. Then the firewall tries to match the packet to the existing state.

Three outcomes are possible:

- State is found but is invalid for the packet. For example, the sequence numbers of TCP packets might not be in the expected range.
- State is found and is valid so the packet is accepted.
- No state is found, in which case the firewall will try to find a matching rule (Rule check).

In the last case, where no state is found, the firewall looks up a list of rules to find a matching rule. If no matching rule is found, the firewall accepts the packet with no further action.

---

**Note -** Figure 2, "Packet Flow in the OpenBSD Packet Firewall," on page 44 does not illustrate the "no further action" case.

---

If the rule is found, then it comes with a policy action, either `block` or `pass`. By default, the `pass` rule in PF creates state for a matching packet. To prevent state creation, use the `no state` option in a particular rule.

If the packet matches the `block` rule, then it is discarded. Both `block` and `pass` actions accept an optional `log` action to log the packet.

The `pass` rule might add optional NAT actions and routing actions for the packet.

- NAT actions change the source address, the destination address, or both addresses.
- Routing actions, called policy-based routing (PBR), select the outbound NIC for forwarding the packet to its destination. PBR can override two parameters that the IP module uses to route packets, the outgoing interface and the next hop address. Otherwise, the IP module makes all routing decisions.

  PBR is more flexible than routing in the IP module. The IP module's routing table uses only the destination address to decide which network interface to use to forward a packet. The firewall can use virtually any field in a packet header when determining which network interface should forward a packet.

As shown in Figure 1, "OpenBSD Packet Firewall," on page 42, when an accepted packet leaves the firewall, an inbound packet travels to the IP module and an outbound packet travels to the appropriate NIC (TX).

## Packet Filter Configuration File and the `firewall` Service

PF uses the `pf.conf` file for all firewall configuration information. If you do not supply a `pf.conf` file at installation, Oracle Solaris provides the "Default Rule Set From the `firewall` Package" on page 46. The service then transitions to the `degraded` state, because this configuration provides no network protection. This state indicates that the administrator must properly configure the system's firewall policy.

The `firewall` service loads a `pf.conf` file as follows:

1. The `start` method calls the `pfctl` command to load the `pf.conf` file from location specified in the `firewall/rules` property.

   To list `firewall` service property values, see "How to Monitor the PF Firewall on Oracle Solaris" on page 71.
2. If the load succeeds, the method runs the `pfctl -e` command to enable the firewall.
3. If `pf.conf` fails to load, for example, due to a syntax error or a missing file, the method enables the firewall with the "Basic Protection Rule Set" on page 47.

The default location of the PF configuration file is `/etc/firewall/pf.conf` and the file contains:

- `set` directives that tune various PF firewall parameters, such as timeouts, debug level, and IP fragment reassembly. See the `set` command in the `OPTIONS` section in the `pf.conf`(7) man page.
- Firewall rules that set your network policy. For more detail, see "Packet Filter Rule Syntax" on page 48.

  For sample rules, see "Examples of PF Rules Compared to IPF Rules" on page 54 and "Packet Filter Macros, Tables, and Interface Groups" on page 52.

See also the `pf.conf`(7) and `pfctl`(8) man pages.

## Default Rule Set From the `firewall` Package

The `/etc/firewall/pf.conf` PF configuration file that the `firewall` package installs is similar to the following:

```
## PF does IP reassembly by default.
# On Oracle Solaris, the 'no-df' option ensures that IP reassembly works
# with broken stacks that send packets with the invalid flag combination 'MF|DF'.
#
```

```
set reassemble yes no-df

#
# PF should not filter loopback traffic by default.
#
# Filtering on loopback can interfere with zone installation and other
# operations due to Oracle Solaris loopback optimizations.
# See the pf.conf(7) man page for guidance on how to enable it
# for your application.
set skip on lo0
##
```

This initial configuration provides no protection from network threats. If you enable the `firewall` service with this default configuration, the service transitions to the `degraded` state, because this configuration provides no network protection. This state indicates that the administrator must properly configure the system's firewall policy. For more information, see the `pf.conf(7)` man page.

## Basic Protection Rule Set

Whenever the default firewall instance fails to start due to a misconfiguration, such as a syntax error in the configuration file or a non-existent configuration file, the service instance is put into the `maintenance` state. In this case, to ensure at least basic network security, the `start` method loads the following basic protection rule set:

```
## basic protection rule set
         # ignore traffic travelling within loopback
         set skip on lo0

         # block everything unless told otherwise and send TCP-RST/ICMP
         # unreachable for every packet which gets blocked
         block return

         # accept incoming SSH connections
         pass in proto tcp to any port 22

         # allow DHCP do its work - incoming messages
         pass in inet proto udp from port 67 to port 68
         pass in inet6 proto udp from port 547 to port 546

         # packet too big - needed for PMTUD
         pass in inet6 proto ipv6-icmp icmp6-type 2

         # router advertisement
```

```
pass in inet6 proto ipv6-icmp icmp6-type 134

# neighbor solicitation
pass in inet6 proto ipv6-icmp icmp6-type 135

# neighbor advertisement
pass in inet6 proto ipv6-icmp icmp6-type 136

# allow all connections initiated from this system, this
# includes e.g. DHCP requests
pass out
```

## Packet Filter Rule Syntax

The syntax of PF rules is deceptively similar to IPF syntax:

*action match-parameter optional-action-1 optional-action-2 ...*

However, the results of identical rule interpretation can be very different. IPF rules do not translate easily into PF rules.

For example, the following rule set is valid in PF and IPF. However, IPF lets client `198.51.100.2` connect to `203.0.113.2` using Secure Shell, while PF does not. The explanation of different behavior is explained in "Differences Between PF and IPF in State Matching" on page 55.

```
block in from 203.0.113.2 to 198.51.100.2
block in from 198.51.100.2 to 203.0.113.2
pass in proto tcp from 198.51.100.2 to 203.0.113.2 port = 22 keep state
```

**Note -** By default, any packet that does not match any rule in the configuration file is accepted by the firewall.

A rule in PF uses actions, match parameters, and optional actions to process packets and determine whether they are accepted or dropped. PF applies the action to the packet if the packet matches the rule. You write a rule by using the following elements in order:

1. Begin the rule with an action. For a list, see "Packet Filter Rule Actions" on page 49.
2. Match desired parameters. For a list, see "Packet Filter Rule Match Parameters" on page 49.
3. Include desired optional actions. For a list, see "Packet Filter Rule Optional Actions" on page 51.

For the complete grammar and syntax of PF rules, see the `pf.conf(7)` man page. For examples of rules whose policy or syntax differs between IPF and PF rules, see "Examples of PF Rules Compared to IPF Rules" on page 54.

## Packet Filter Rule Actions

Each rule begins with an action. PF applies the action to the packet if the packet matches the rule. The following list includes the commonly used actions applied to a packet and indicates whether the action was also in IP Filter.

anchor
: Only in PF. Opens a new rule set which should be further applied to a matching packet.

`block`
: Prevents the packet from passing through the filter.

```
## without an in, out, or on match parameter,
## blocks all traffic on all interfaces
block all
```

`pass`
: Allows the packet through the filter. Applies to all packets, incoming and outgoing.

```
pass all
```

`match`
: Provides fine-grained filtering without altering the `block` or `pass` state of a packet.

`match` rules differ from `block` and `pass` rules in that the parameters are set every time a packet matches the rule, not only on the last matching rule. These sticky parameters are in effect until explicitly overridden. Parameters affected are `nat-to`, `binat-to`, `rdr-to`, and `scrub`.

## Packet Filter Rule Match Parameters

Keywords in this list define criteria that determine whether a packet matches the rule.

`in`
: Applies the rule only if rule matches an inbound packet.

```
pass in from any to any port = 22
```

`out`

Applies the rule only if the rule matches an outbound packet.

```
pass out log on net0
```

`on` *interface-name*

Matches a packet that is moving in or out of the specified interface.

```
pass out log on net0
```

`from` *source* `to` *destination*

Applies the rule only if the rule matches a packet from the specified source.

A source or a destination can be one of the following:

- An IP address.

  CIDR notation is accepted, for example, `198.51.100.0/27`.
- A reference to a table, such as a whitelist of approved email sources.
- A network interface name such as `net0`, which gets expanded to list of IP addresses and those addresses are plumbed to the interface.

The following example illustrates that well-formed rules do not require the use of `in`, `out`, or `on`. This rule matches all inbound and outbound packets on any interface on the host.

```
pass from any to any
```

The `any` keyword is used to accept packets from all sources and to all destinations. The IP addresses can be modified by a port number. The following example rule matches every inbound packet coming to port 22.

```
pass in from any to any port = 22
```

`flags` *a*/*b* `| any`

Matches TCP packets based on the TCP flags that are set.

The rule applies only to TCP packets that have the flags *a* set out of set *b*. Flags not in *b* are ignored.

The flags are: `(F)IN`, `(S)YN`, `(R)ST`, `(P)USH`, `(A)CK`, `(U)RG`, `(E)CE`, and `C(W)R`.

The following examples describe some sample flags settings:

- `flags S/S` – Flag `S` is set and the other flags are ignored.
- `flags any` – No flags are checked.
- `flags S/SA` – Flag `S` is set. Default setting for stateful connections.

  `SYN` and `ACK` together cannot match a packet. `SYN+PSH`, `SYN+RST`, and `SYN` can match a packet, but `SYN+ACK`, `ACK+RST`, and `ACK` cannot.
- `flags /SFRA` – If the *a* set is not specified, it defaults to none. All *a* flags must be unset before using this filter.

| | |
|---|---|
| | See the `pf.conf(7)` man page for further uses and consequences of flags settings. |
| `icmp-type` | Matches the packet based on ICMP type. This keyword is used only when the `proto` option is set to `icmp` and is not used if the `flags` option is used. |
| `proto` | Matches a specific protocol. You can use any of the protocol names specified in the `/etc/protocols` file, or use a decimal number to represent the protocol. The keyword `{tcp, udp}` matches either a TCP or a UDP packet. |
| `tos` | Matches the packet based on the type-of-service value expressed as either a hexadecimal or a decimal integer. |
| `ttl` | Matches the packet based on its time-to-live value. A packet's stored time-to-live value indicates the number of hops the packet can make before being discarded. |
| `group` | Matches incoming packets by the effective group ID. |
| `user` | Matches incoming packets by the effective user ID. |

## Packet Filter Rule Optional Actions

Keywords in this list define additional optional actions.

| | |
|---|---|
| `allow-opts` | When specified for a `pass` rule, allows packets to pass the filter based on the last matching rule even if the packets contain IP options or routing extension headers. Without `allow-opts`, PF blocks IPv4 packets with IP options and IPv6 packets with routing extension headers. |
| `keep` *keep-options* | An abbreviation for `keep state`. Determines the information that is kept for a packet that matches a given rule. Because the `pass` rules in PF create a state by default, the `keep` option is useful to further specify options for the kept state, such as `sloppy` and `if-bound`. |
| `nat-to` | Specifies that IP addresses are to be changed as the packet traverses the given interface. This technique allows one or more IP addresses on the translating host to support network traffic for a larger range of systems |

| | |
|---|---|
| | on an internal network. Internal network addresses should conform to the address ranges defined in Address Allocation for Private Internets, RFC 1918. |
| `rdr-to` | Redirects the packet to another destination and possibly a different port. `rdr-to` can specify port ranges as well as single ports. |
| `log` | Logs the matching packet. If `log` is used in a `match` action, then the packet is logged immediately on `match` even if the state is not kept. Multiple matching `match` rules will log the packet multiple times.<br><br>`log` accepts the following arguments:<br>■ `(all)` to log all packets regardless of state<br>■ `(matches)` to log the packet on all subsequent matching rules<br>■ `(user)` to add UID and PID socket information to the log<br>■ `(to` *interface*`)` to send logs to a specified capture interface |
| `quick` | Executes the rule containing the `quick` option if the packet matches the rule. All further rule checking stops. |
| `route-to` | Moves the matching packets to an outbound queue on a named interface. Additionally, `route-to` can send different packets to different interfaces or different next hops by way of specified interfaces. Implements policy-based routing (PBR).<br><br>`route-to` accepts the following arguments. Note that the (*interface-name* + *next-hop* address) pairs have two syntax variants:<br>■ *interface-name*<br>■ `(`*interface-name next-hop-address*`)` or *next-hop-address@interface-name*<br>■ `{ (`*interface1-name next-hop1-address*`), (`*interface2-name next-hop2-address*`) [, ...] }` or `{` *next-hop1-address@interface1-name, next-hop2-address@interface2-name*`[, ...] }`<br><br>For how to select which interface to use when specifying several interfaces, see `http://www.openbsd.org/faq/pf/pools.html`. |

## Packet Filter Macros, Tables, and Interface Groups

PF provides macros, tables, and interface groups to ease the readability, extensibility, and reuse of PF rules. Macros and tables accept lists. Interface groups simplify applying the same policy to multiple systems where the interface names that handle the policy are different on those systems.

- Macro – Defines one or more items to be treated as one item. Specify a name that is easy to understand. For example, the following rule uses two easily understood macros:

  ```
  emailserver = 192.1.2.223
  email = "{ smtp, pop3, imap }"  ## mail services
  pass in from any to $emailserver port = $email
  ```
- Table – Lists of IP addresses that can be manipulated without reloading the entire rule set. Promotes fast lookups.

  A table is defined by a rule or set of rules. If all rules which refer to a table are deleted, the table is destroyed. To create a table that would outlive its rules, use the `persist` keyword.

  For example, the following `clients` table is a list of clients. One address in the client range is disallowed:

  ```
  table <clients> persist { 198.51.100.0/27, !198.51.100.5 }
  ```
- Group – Name for a group of interfaces. The group name is used in PF configuration rules. Any policy that is defined for the group is then applied to interfaces that are members of the group. A group is also known as a *firewall interface group*, where the members of the group handle similar types of network traffic. The group name can be any string of up to 31 characters, starting with an alphabetic character and not ending in a digit.

  Administrators add interfaces to the groups and use the group name in rules. After the administrator adds interfaces to groups using the `ipadm` command and loads rules referring to the group, PF can use the group name to match packets. An interface can be a member of several interface groups. For an example, see Example 11, "PF Configuration File Using Firewall Interface Groups," on page 76.

**EXAMPLE 3** NAT Rule in PF

This example illustrates how to construct a NAT rule. This rule translates the source address in the outbound packet to an address which is bound to the `net2` interface. It rewrites a packet that goes out on the `net2` device with a source address of `198.51.100.0/27` and externally shows its source address as `net2`:

```
## NAT rule that externally shows net2 as source address
pass out on net2 from 198.51.100.0/27 to any nat-to (net2)
```

**EXAMPLE 4** Spam Rule in PF

This example illustrates the use of tables in PF. The administrator creates a `spam` table for IP addresses that send spam. The following firewall rule blocks incoming packets from all addresses in the `spam` table.

```
table <spam> { 198.51.100.0/27 }
block in from <spam> to any
```

To block a new spam source, the administrator updates the table only.

```
# pfctl -t spam -T add 203.0.113.0/16
```

The `pfctl` command updates the firewall configuration without altering the rule set. In the kernel, the table now reads:

```
table <spam> { 198.51.100.0/27 203.0.113.0/16 }
```

For the complete grammar and syntax of PF rules, see the `pf.conf`(7) man page. For arguments to the `ipadm` command, see the `ipadm`(8) man page.

## Examples of PF Rules Compared to IPF Rules

PF and IPF have different default filtering behavior but use a similar syntax. Note that unmodified IPF rules in the PF configuration file are likely to implement the wrong security policy. The following examples illustrate some of the differences. For ways to verify your PF rules, see "Using PF Features to Administer the Firewall" on page 61.

### Loopback Interface Filtering Is On by Default in PF

Unlike IPF, PF processes the packets that are bound to the loopback interface by default. Consider the following rule set:

```
pass out all
block in all
```

The `pass out all` rule enables the system to connect to remote servers. Incoming packets from remote clients get dropped by the `block in all` rule. Because this rule set also applies to loopback traffic, the system is effectively prevented from connecting to local servers. For example, `ssh localhost` fails with the error message, `connection time out`.

## Differences Between PF and IPF in State Matching

PF and IPF match particular packets to state in different ways. Consider the following rule set on a router that forwards traffic between the `198.51.100.0/27` and `203.0.113.0/16` networks:

```
## Valid rule set in IPF and PF

block in from 198.51.100.0/27 to any
block in from 203.0.113.0/16 to any
pass in from 198.51.100.2 to 203.0.113.2 keep state
```

Although these rules are valid for both firewalls, they implement different policies.

- In IPF, the `198.51.100.2` client can reach the `203.0.113.2` server and the server's response packets can reach the client.

  1. The `block` rules prevent all traffic between the `198.51.100.0/27` and `203.0.113.0/16` networks.
  2. The `pass` rule allows the `198.51.100.2` client to connect to the `203.0.113.2` server.
  3. The `keep state` action in the `pass` rule allows the server's responses to reach the client.

  As soon as the first request packet sent by `198.51.100.2` matches the `pass` rule, a state is created: `198.51.100.2 -> 203.0.113.2`. The state also matches the reverse packets sent by the server back to the client: `203.0.113.2 -> 198.51.100.2`.
- In PF rule processing, the `198.51.100.2` client cannot connect to the `203.0.113.2` server.

  1. PF state inspection is stricter. The `in` match parameter to the `pass` rule instructs PF to match only inbound packets. Therefore, the state that the `pass in` rule creates matches only two kinds of packets:
     - Inbound forward packet, sent by client to server: `198.51.100.2 -> 203.0.113.2`
     - Reverse outbound packet, sent by server to client: `203.0.113.2 -> 198.51.100.2`
  2. When a response arrives from the server to the PF firewall, PF does not see the packet as a reverse packet but as inbound for the first time, so the packet does not match the state that the `pass in` rule creates. Rule processing continues to look for a rule that matches the packet to determine whether to forward the packet or drop it. The packet matches the second `block` rule, `block in from 203.0.113.0/16 to any`, so PF drops the response sent by the server.

  For the traffic to be routed in PF as it is in IPF, you have two options:

  - Add a rule that creates a state for forward outbound packets. This state will enable the inbound reverse packets from the server to be valid.

    ```
    ## PF rule set1 enforces IPF rule set policy
    ```

```
block in from 198.51.100.0/27 to any
block in from 203.0.113.0/16 to any
pass in from 198.51.100.2 to 203.0.113.2 keep state
pass out from 198.51.100.2 to 203.0.113.2 keep state
```

The `pass` rules create two states:

```
198.51.100.2 -> 203.0.113.2 @ in
198.51.100.2 -> 203.0.113.2 @ out
```

These states allow reverse inbound packets from the server:

```
in: 203.0.113.2 -> 198.51.100.2
```

- Write a simpler rule set that does not use packet direction:

  ```
  ## PF rule set2 enforces IPF rule set policy

  block in from 198.51.100.0/27 to any
  block in from 203.0.113.0/16 to any
  pass from 198.51.100.2 to 203.0.113.2 keep state
  ```

  The `pass` rule without an `in` parameter matches packets sent by the client regardless of direction. Therefore, when PF intercepts a packet from `198.51.100.2` as inbound for the first time, it creates a state:

  ```
  198.51.100.2 -> 203.0.113.2 @ in
  ```

  After the packet is routed by the firewall, it is intercepted by PF again, this time as an outbound packet. When the packet hits the `pass` rule, PF creates a second state:

  ```
  198.51.100.2 -> 203.0.113.2 @ out
  ```

## Network Address Translation in PF

The OpenBSD version of PF supports NAT-64 as described by RFC 6146, while the Oracle Solaris version of PF supports traditional NAT only, as described by RFC 2663.

NAT typically sets up mapping between a network with a private address range and the Internet. NAT enables hosts in private networks to talk to any host on the Internet. In a typical deployment, the mapping is one to many, which means that many hosts in a private network share one public IP address to connect to Internet servers.

In PF, NAT processing is an optional rule action. A NAT rule in PF effectively creates a state that tells the firewall to alter the IP address depending on the action type of either `rdr-to` or `nat-to`, and the matching packet's direction, either inbound or outbound.

In PF, the `nat-to` and `rdr-to` actions are executed as soon as a packet matches the rule with the `nat-to` or `rdr-to` optional action. The subsequent rules see an already-translated packet. Both actions also create state for the matching packet.

- `nat-to` – Typically used for outbound packets. Allows a network client in a private network to talk to an Internet server. `nat-to` tells the firewall to overwrite the source address and port in the matching packet according to additional parameters in the rule.

  For example, the following rule changes a private source address from the address range `198.51.100.0/27` in a packet to a public IP address that is bound to the `net0` interface:

  ```
  pass out on net0 from 198.51.100.0/27 to any nat-to (net0)
  ```
- `rdr-to` – Typically used for inbound packets. The `rdr-to` optional action is the opposite of `nat-to` in that it allows a client on the Internet to talk to a server behind the PF firewall in a network with private IP addresses. `rdr-to` changes the destination address in a matching packet.

  For example, the following rule redirects inbound SMTP traffic to `203.0.113.2:2525`, the IP address of the SMTP server on a private network behind the firewall. This rule matches any TCP inbound packet coming to the `net0` interface and having the same destination IP address as the address that is bound to `net0`, and whose destination port is 25. The `rdr-to` action overwrites the destination address and port in the packet with `203.0.113.2:2525`.

  ```
  pass in on net0 inet proto tcp from any to (net0) port = 25 rdr-to 203.0.113.2 port
   2525
  ```

Changes from the `rdr-to` or `nat-to` actions are immediately visible to subsequent rules unless the affected packet does not match a `pass` or `block` rule that creates a state. In that case, the packet that leaves PF is not touched by the `rdr-to` or `nat-to` option. A `log` action still logs the packet.

```
## Block packets going to 198.51.100.2
match out to 198.51.100.2 nat-to 198.51.100.1
block from 198.51.100.1
```

The effect of the preceding example is that the packets sent by the `*2` address hit the block rule because they are matched to `*1`, which is blocked. If the `match out to` rule were not in the rule set, the `*2` packets would go through.

### Rule Equivalents Using `match` and `pass` Actions

All of the parameters of `match` rules that require a `pass` or `block` rule create a state. Sometimes a `pass` rule can be equivalent to `match` plus `pass` rules. For example, the following PF rule sets are equivalent:

```
webserver = "198.51.100.7"
webports = "{ http, https }"
emailserver = "198.51.100.5"
email = "{ smtp, pop3, imap }"

### Rule set 1 - match action then pass action
match in on $ext_if proto tcp to $ext_if port $webports rdr-to $webserver
match in on $ext_if proto tcp to $ext_if port $email rdr-to $emailserver

pass proto tcp from any to $webserver port $webports
pass proto tcp from any to $emailserver port $email
pass proto tcp from $emailserver to any port smtp

### Rule set 2 - no match action, only pass action
pass in on $ext_if inet proto tcp to $ext_if port $webports rdr-to $webserver
pass in on $ext_if inet proto tcp to $ext_if port $email rdr-to $mailserver
pass on $int_if inet proto tcp to $webserver port $webports
pass on $int_if inet proto tcp to $mailserver port $email
```

## Packet Filter Rule Processing

PF processes the rules according to a "last match" policy, which means that the policy decision on a packet is determined by the last rule that matched the packet. This policy suggests that rules are best ordered from generally applicable rules first to more detailed match parameters later in the rule set.

For example, consider the following rule set:

```
block in from any to any
pass in from any to any port = 22
pass in from any to any port = 25
```

The first rule blocks all inbound packets. The following rules match a destination port number.

So, if a packet comes to port 22, PF processes it as follows:

1. Applies the block rule.

2. Applies the port 22 rule, which matches the packet. PF keeps the match in memory.
3. Applies the port 25 rule, which does not match the packet.

After PF checks the packet against all rules, it forwards the packet due to the success of the port 22 rule.

The `quick` keyword is an exception. If a rule includes the `quick` keyword, the action for that rule is executed immediately and no further rules are applied to that packet.

```
block in from any to any
pass in quick from any to any port = 22
pass in from any to any port = 25
```

If you pass the same packet through the firewall and use this rule set, PF executes the `pass` action on the packet immediately when the second rule is applied. The `quick` keyword stops further checking of that packet, so the third rule is not applied.

For more information, see the `pfctl`(8) and `pf.conf`(7) man pages.

## Packet Filter Logging

The PF log daemon, `pflogd`, writes packets that PF sends to a capture datalink to a log in `libpcap` binary file format. Logging is enabled by the `log` action per rule and is optional. For log options, see `log` entry in "Packet Filter Rule Optional Actions" on page 51.

The `pflogd` daemon runs as the `svc:/network/firewall/pflog` SMF service. By default, the `log` action sends packets to the `pflog0` datalink. The packets are written by the `pflog:default` service instance to a `pflog0.pkt` log file in the `/var/log/firewall/pflog` directory.

The `pflog` service adds selective filtering to PF's default logging:

- PF sends packets that are logged due to a `log` action to the specified capture link. If the action does not specify a link, PF uses `pflog0` by default.
- Packets that are intercepted at the capture link can be further filtered by BPF (Berkeley Packet Filter). This filtering is configured by a userland application such as `pflog` or `tcpdump` or Wireshark, to select just a subset of the captured packets for logging.

  Because the BPF filter drops unwanted captured packets in the kernel, filtering saves CPU cycles. Those packets are not copied to userland.

For ways to customize packet logging, see "Using Packet Filter Logging" on page 65 and the `pflogd`(8) man page.

## Packet Filter References

The following PF man pages are written by or modified for Oracle Solaris:

| | |
|---|---|
| `firewall`(7) | Describes the `firewall` service. |
| `pfctl`(8) | Manages PF rules, displays tunables, and performs other tasks. |
| `pf.conf`(7) | Stores packet filtering rule sets. |
| `pf.os`(7) | An operating system (OS) fingerprint database that stores OS signature information. |
| `pflogd`(8) | Maintains a log of PF actions. |
| `ftp-proxy`(8) | Handles FTP transfers that pass through PF doing NAT |

The following references provide additional explanations, examples, tutorials, and man pages.

- Hansteen, Peter N.M. The Book of PF, 3rd Edition. No Starch Press. ISBN 978-1-59327-589-1, 2014.
- PF: The OpenBSD Packet Filter (`http://www.openbsd.org/faq/pf/index.html`)
- OpenBSD Man Page Search (`http://man.openbsd.org`)

♦ ♦ ♦ C H A P T E R 5

# Configuring the Firewall in Oracle Solaris

This chapter provides instructions for implementing a firewall using OpenBSD Packet Filter (PF) software. For overview information, see Chapter 4, "Oracle Solaris Firewall".

This chapter covers the following topics:

## Using PF Features to Administer the Firewall

`pfctl` is the PF command-line tool for managing the PF kernel driver. This section provides examples of using `pfctl` to administer firewall.

The `pfctl` command can display the syntax of the rules and order of execution, verify the validity of the configuration file, and perform other useful administration tasks.

The following examples assume that you have been assigned the Network Firewall Management rights profile and that you are running the `pfctl` commands in a `pfbash` shell.

- Use the `-sr` options to display the main, or root, rule set.

  ```
  $ pfbash
  $ pfctl -sr
  ```
- Use the `-a` and `-sr` options to display the complete rule set from the root to the leaf.

  ```
  $ pfctl -a '*' -sr
  ```
- Use the `-sA` option to display all anchors in the rule set tree.

  ```
  $ pfctl -sA anchor
  ```

- Use the `-n` option to check the syntax of a rule file without loading the rules into the kernel. For example, the following command checks the syntax of the rules in the `pf.conf` file in the `/etc/firewall/test` directory.

  ```
  $ pfctl -nf /etc/firewall/test/pf.conf
  ```
- Use the `-x` option to set the debugging level. The default debugging level is `error`.

  ```
  $ pfctl -x debug
  # dmesg
  ```

  The debug messages print to the console only. The `dmseg` command finds recent diagnostic messages in the system buffer and prints them to standard output.
- Use the `-g` option to debug problems with the rule parser.

  ```
  $ pfctl -g -n -f /etc/firewall/test/pf.conf
  ```
- Use the `-v` and `-vv` options to display verbose output.

  ```
  $ pfctl -vv
  ```
- Use the `-r` option to perform DNS lookups on states when displaying them.

  ```
  $ pfctl -r -sr
  ```

For more options, see the `pfctl`(8) man page.

## Preparing to Configure the Oracle Solaris Firewall

This section describes the IPF to PF conversion tool, how to configure the `svc:/network/firewall/pflog` service for logging PF packets, and how to configure the `ftp-proxy` service, which enables the FTP protocol to work when NAT is performed by the firewall. You should configure the `pf.conf` network policy file, the logging options, and FTP handling before enabling the `firewall` service.

### IP Filter to Packet Filter Rules Conversion Tool

To assist you in converting IPF rules to PF, the Oracle Solaris 11.4 release provides the `ipf2pf` service, which only runs at the first boot to a fresh BE. The service stores the obsolete IPF configuration files in the `/var/firewall/legacy.ipf/conf` directory. The service then converts the IPF configuration to PF. The migration process intentionally requires manual intervention to force you, the administrator, to configure and verify your network policy.

The conversion is written to the `/var/firewall/legacy.ipf/conf/pf.conf` file. Although the converted configuration will work, you should treat it as a suggestion. As the conversion examples illustrate, the result can be suboptimal. However, optimization can introduce errors, so consider the following when converting your IPF rules to PF:

- Do not rely solely on the `ipf2pf` tool to create your firewall policy
- Use your brain and the network policy specification
- Check everything twice before altering firewall rules
- Back up every step so that you can retrieve a working configuration

When you are satisfied that the PF rule set created by the `ipf2pf` service plus your modifications is accurate, move the resulting `pf.conf` file to `/etc/firewall/pf.conf`.

The PF rules suggested by the `ipf2pf:default` service instance are annotated by comments to explain the differences from IPF. The core part of the service is the `/usr/bin/ipf2pf` command, which performs the conversion.

The following simple examples illustrate the tool. For more complex examples, see "Examples of Converting IPF Rules to PF Rules" on page 77.

**EXAMPLE 5** Converting All IPF Configuration Files to PF

Unlike PF, IPF uses several configuration files. The `ipf2pf` can generate one annotated PF configuration file that combines all the IPF files:

```
$ pfbash ipf2pf -4 /etc/ipf/ipf.conf  -6 /etc/ipf/ipf6.conf \
      -n /etc/ipf/ipnat.conf -p /etc/ipf/ippool.conf \
      -o /var/firewall/legacy.ipf/conf/mypf.conf
```

---

**Tip -** If you plan to use separate configuration files similar to IPF files, add an `INCLUDE` statement to the PF configuration file for the main (root) rule set.

```
$ pfedit /etc/firewall/pf.conf
...
include "/etc/firewall/pfzones.conf"
```

---

**EXAMPLE 6** Simple IPF to PF Conversion of IPv4 Rules

You can try the tool on your own rule samples. Consider the following IPv4 rules:

```
    ## IP Filter Sample ipf.conf File
    # Stateful firewall allows everything outbound
    # and blocks everything inbound except ssh
    #
    pass out all keep state
    block in all
    pass in from any to any port = 22 keep state
```

When you run the `ipf2pf`command on the `/etc/ipf/ipf.conf` file, the command produces the following output:

```
$ ipf2pf -4 ipf.conf

    #
    # File was generated by ipf2pf(7) service during system upgrade. The service
    # attempted to convert your IPF rules to PF (the new firewall) rules. You
    # should check if firewall configuration here, suggested by ipf2pf, still meets
    # your network policy requirements.
    #

    #
    # Unlike IPF, PF intercepts packets on loopback by default.
    # IPF does not intercept packets bound to loopback. To turn off the
    # policy check for loopback packets, we suggest to use command
    # below:
    set skip on lo0
    #
    # PF does IP reassembly by default. It looks like your IPF
    # does not have IP reassembly enabled. Therefore the feature is
    # turned off.
    #
    set reassemble no
    # In case you change your mind and decide to enable IP reassembly
    # delete the line above. Also to improve interoperability
    # with broken IP stacks, tell PF to ignore the 'DF' flag when
    # doing reassembly. Uncommenting line below will do it:
    #

    # set reassemble yes no-df

    block drop in inet all
    #
    # IPF rule specifies either a port match or return-rst action,
    # but does not specify a protocol (TCP or UDP). PF requires a port
    # rule to include a protocol match using the 'proto' keyword.
    # ipf2pf always assumes and enters a TCP port number
    #

    pass in inet proto tcp from any to any port = 22 flags any keep state (sloppy)
    pass out inet all flags any keep state (sloppy)
```

For more information, see the `ipf2pf(7)` and `firewall(7)` man pages.

## Using Packet Filter Logging

Logging packets is an additional SMF service in PF. To log packets, you need to enable the logging service and add log actions in your PF rules. The default logging service instance is `pflog:default`.

The following describes the overall process:

1. A `pflog` service instance creates a capture datalink. The datalink is stored as the value of the service property `interface`.
2. A packet enters PF and matches a rule with a `log` action.
3. PF sends the matched packet to the capture datalink specified by the `log` action. If no capture datalink is specified, PF uses the default (`pflog0`) capture datalink.
4. The `pflog` service instance that reads the capture datalink might attach a BPF filter to the capture link, similar to what `tcpdump` might do. The filter selects a subset of the packet to log. If no filter expression exists, which is the default, all packets are sent to the `pflog` daemon and then copied into the log.
5. The administrator archives the log files, refreshes the `pflog` service to restart the logging, and moves the log files to another system for inspection.

   The log file written by the `pflog` daemon is in `libpcap` format. Oracle Solaris provides several tools that can read this format, including `tshark`, `tcpdump`, and Wireshark. For more information, see the `tcpdump`(8), `tshark`(1), and `pcap`(3PCAP) man pages.

---

**Tip -** For ease of configuration and debugging, name your capture datalink and log file identically by ending `pflog` service instance names with a number, such as `pflog1` or `pflog21`.

---

Initially, packet logging uses the `pflog:default` service instance, which sends packets to the `pflog0.pkt` log file. You might want to create new service instances for packets that arrive on a particular port, or for packets that need a longer `snaplen`, or for debugging or other purposes. For the properties that you can specify for a `pflog` service instance, see the `pflogd`(8) man page.

To see the initial values of the default `pflog` service properties, run the following command on a newly-installed system:

```
$ svcprop -g application pflog
pflog/delay integer 60
pflog/filter astring ""
pflog/interface astring pflog0
pflog/logfile astring /var/log/firewall/pflog/pflog0.pkt
pflog/snaplen integer 160
```

The following examples illustrate typical log file administration tasks. For more examples, see the `pflogd(8)` man page.

**EXAMPLE 7** Creating a New `pflog` Service Instance

The following command creates a new instance called `pflssh1` for logging packets whose source or destination is port 22.

```
$ pfbash pflogd -C pflssh1 port 22
```

The following command shows the configuration of the instance, including its log file and filter.

```
$ pflogd -c pflssh1
PF pflogd configuration:
        - logfile:
                /var/log/firewall/pflog/pflssh1.pkt
        - snaplen:
                160
        - interface:
                pflssh1
        - delay:
                60
        - filter:
                port 22
```

When a packet matches a PF rule that includes the `log (to pflssh1)` action and `port 22`, the `pflogd` daemon adds a log entry to the `pflssh1.pkt` file from the capture datalink.

For example, packets that match the following rule become entries in the `pflssh1.pkt` file:

```
pass in log (to pflssh1) proto tcp to any port 22
```

Note that packets from a remote host to another remote host's port 22 will be logged.

Similarly, as an example of selective filtering, packets from or to port 22 will be logged by this rule:

```
pass in log (to pflssh1) proto tcp from any to any
```

**EXAMPLE 8** Specifying a Log File for a `pflog` Service Instance

In this example, the logs of the new service instance are placed in the `debug/debug0.pkt` directory below the `/var/log/firewall/pflog/` directory.

```
$ pfbash pflogd -C debug0 -f /var/log/firewall/pflog/debug/debug0.pkt
```

**EXAMPLE 9** Rotating PF Log Files

The following command archives the current log. After the administrator refreshes the service, the `pflog0.pkt` log file is empty and ready for new entries.

```
$ pfbash mv /var/log/firewall/pflog/pflog0.pkt /var/log/firewall/pflog/
archive1pflog0.pkt
$ svcadm refresh pflog:default
```

The administrator copies the archived files to another system for inspection.

```
# scp /var/log/firewall/pflog/archive*.pkt username@192.0.2.44:/logs/pflogs/
```

**Caution -** Do not store packets from different `pflog` instances in the same log file. And, do not have more than one instance use the same capture datalink simultaneously.

## Using the `ftp-proxy` Service

Without additional support, FTP transfers that pass through PF doing NAT would be broken. The `ftp-proxy` service provides this support. PF redirects the FTP packets to an address where the `ftp-proxy` service listens, which is `127.0.0.1` by default. You specify the address and port where the proxy listens for PF's redirected FTP connection attempts. The proxy negotiates the connection on behalf of the client.

The FTP proxy is transparent to the client by altering the firewall configuration at runtime according to the FTP rules that you provide. These rules, that contain the address and port where the proxy listens, follow an anchor, such as `"ftp/*"`, in the `pf.conf` file.

Before enabling the service, you also ensure that the `ftp-proxy` service rules can be processed by PF. For the steps, see "How to Make FTP Transfers Pass Through PF Doing NAT on Oracle Solaris" on page 69.

## Configuring the Packet Filter Service on Oracle Solaris

The following procedures configure PF on an Oracle Solaris system:

- "How to Configure the Firewall on Oracle Solaris" on page 68
- "How to Monitor the PF Firewall on Oracle Solaris" on page 71

- "How to Make FTP Transfers Pass Through PF Doing NAT on Oracle Solaris" on page 69
- "How to Use Groups to Simplify Firewall Policy in a Network" on page 73

## ▼ How to Configure the Firewall on Oracle Solaris

To run PF as your firewall, you configure the `pf.conf` file to reflect your policy, then enable the `firewall` service. To log PF events, see "Using Packet Filter Logging" on page 65.

**Before You Begin** To configure the `firewall` service, you must become an administrator who is assigned the Network Firewall Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Install the PF package.**

   Perform this step if you did not install a group that contains the PF package. The `solaris-small-server`, `solaris-large-server`, and `solaris-desktop` group packages include the PF package. You must become an administrator who is assigned the Software Installation rights profile.

   ```
   $ pfbash pkg install firewall
   ```

2. **Create or update your packet filtering rule set and verify the syntax.**

   ```
   $ pfedit /etc/firewall/pf.conf
   $ pfctl -nf /etc/firewall/pf.conf
   ```

   For sample rules, see "Packet Filter Macros, Tables, and Interface Groups" on page 52 and "Examples of PF Rules Compared to IPF Rules" on page 54.

   ---

   **Note -** If you are using a service, such as `ftp-proxy`, you need to add an anchor entry, such as `anchor "ftp/*"`, at an appropriate place in your `pf.conf` file.

   ---

3. **Enable PF.**

   ```
   $ svcadm enable firewall
   ```

   - If the PF configuration file is empty and you enable the `firewall` service, some traffic filtering occurs. For example, PF drops TCP packets with invalid flag combinations.
   - If you provide an invalid `pf.conf` file before enabling the service, the `firewall` service loads the "Basic Protection Rule Set" on page 47, which puts the service into the `degraded` state.

4. **(Optional) Verify that the PF driver is running.**

   The version number is listed in the output.

   ```
   $ modinfo -i pf
   ID  LOADADDR         SIZE   INFO REV NAMEDESC
   244 --               3fdd0  137  1   pf (PF 5.5)
   ```

5. **Enable the `pflog:default` service if you plan to log packets.**

   ```
   $ svcadm enable pflog:default
   ```

   The default location for the log is `/var/log/firewall/pflog/pflog0.pkt`.

   ---

   **Tip -** Schedule regular rotation of PF log files.

   ---

   For examples of configuring packet logging, see "Using Packet Filter Logging" on page 65 and the `pflogd`(8) man page.

6. **(Optional) To disable the service, use the `svcadm` command.**

   ```
   $ svcadm disable network/firewall
   ```

   This command removes all rules from the kernel and disables the service. You might disable the firewall on a system that you have disconnected from the network or that you are decommissioning.

## ▼ How to Make FTP Transfers Pass Through PF Doing NAT on Oracle Solaris

---

**Note -** If you do not use PF to do NAT, you can skip this procedure.

---

In this task, you use the `ftp-proxy` service to make FTP transfers pass through PF doing NAT. You need to specify two addresses:

- The *proxy NAT address* is the source address for connections to FTP servers.
- The *listening address* is where the `ftp-proxy` service listens.

**Before You Begin** To configure the `ftp-proxy` service, you must become an administrator who is assigned the Network Firewall Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Modify the `/etc/firewall/pf.conf` file to enable `ftp-proxy` configuration.**

   Adjust the rules as needed. Depending on the rest of the ruleset, the last rule that explicitly enables FTP sessions from the proxy in the following example may not be necessary.

   ```
   $ vi /etc/firewall/pf.conf
      ...
      anchor "network:firewall:ftp-proxy:default/*"
          pass in quick inet proto tcp to port ftp rdr-to 127.0.0.1 port 8021
          pass out inet proto tcp from (self) to any port ftp
   ```

   If you are not using the default `ftp-proxy` instance, substitute your instance name for `default`, as in:

   ```
      anchor "network:firewall:ftp-proxy:your-instance/*"
   ```

2. **Configure the listening address and the proxy address of the default `ftp-proxy` service, then verify.**

   ```
   $ pfbash ftp-proxy -c default -b 127.0.0.1 -a proxy-nat-address
   $ ftp-proxy -c default
   ```

   The *proxy-nat-address* is the public IP address of the FTP client as seen by the server.

3. **(Optional) Configure the listening address and port, and the proxy address of the anonymous `ftp-proxy` service, then list the configuration.**

   This command uses the default listening address, so does not specify it. Specify a different port if you use the default listening address.

   ```
   $ ftp-proxy -C anonymous -A on -p port -a proxy-nat-address
   $ ftp-proxy -c anonymous
   ```

   If you later decide to allow the `anonymous` instance to handle all FTP connections, turn `anonymous` off.

   ```
   $ ftp-proxy -c anonymous -A off
   ```

4. **Configure your FTP rule set in PF.**

   For example:

   ```
   ## process the ftp-proxy service's rules
   anchor "ftp/*"

   ## redirect ftp connection attempts to the ftp-proxy
     pass in quick inet proto tcp to port ftp rdr-to 127.0.0.1 port 8021

   ## allow outgoing ftp connections from the proxy if needed
   ```

```
  pass out inet proto tcp from (self) to any port ftp

## redirect connections that should be allowed only
## in the anonymous mode to the anonymous ftp-proxy
...
```

5. **(Optional) Log FTP packets or turn off logging that has been enabled.**

   Logged FTP packets are sent to the `pflog0` capture datalink only.

   a. **Add log actions to the FTP rule set.**

   ```
   pass in quick log inet proto tcp to port ftp rdr-to 127.0.0.1 port 8021
   pass out log inet proto tcp from (self) to any port ftp
   ```

   b. **Activate logging for the FTP packets, or stop logging.**

   ```
   $ ftp-proxy -c default -v on      /* equivalent to -v in OpenBSD PF */
   $ ftp-proxy -c default -v all     /* equivalent to -vv */
   $ ftp-proxy -c default -v off     /* stop logging packets */
   ```

   For a description of the three log options, see the `ftp-proxy`(8) man page.

6. **Ensure that the `network/socket-filter:pf_divert` service is online in the global zone.**

   ```
   global-zone $ svcs -x svc://network/socket-filter:pf_divert
   ```

   If the `socket-filter:pf_divert` service is not online in the global zone, enable it.

   ```
   global-zone $ pfbash svcadm enable socket-filter:pf_divert
   ```

7. **Restart the firewall and refresh then enable all `ftp-proxy` service instances.**

   ```
   $ pfbash svcadm restart firewall
   $ svcadm refresh ftp-proxy:default
   $ svcadm refresh ftp-proxy:anonymous
   $ svcadm enable ftp-proxy:default
   $ svcadm enable ftp-proxy:anonymous
   ```

## ▼ How to Monitor the PF Firewall on Oracle Solaris

Monitoring includes viewing firewall service properties, viewing rules as they are running or viewing possible rule sets, and reviewing log files.

**Before You Begin** You must become an administrator who is assigned the Network Firewall Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Examine the status of the `firewall` service.**

   - **Determine whether the Packet Filter service is enabled.**

     ```
     $ svcs -x firewall:default
     svc:/network/firewall:default (Network Firewall)
      State: disabled since Fri Apr 10 10:10:50 2015
     Reason: Disabled by an administrator.
        See: http://oracle.com/msg/SMF-8000-05
        See: pf.conf(7)
        See: /var/svc/log/network-firewall:default.log
     Impact: This service is not running.
     ```

   - **List the configuration file names and locations for the Packet Filter service.**

     ```
     $ svcprop firewall:default | grep ^firewall
     firewall/default_rules_sha256 astring 7734b...bbb
     firewall/fingerprints astring /etc/firewall/pf.os
     firewall/rules astring /etc/firewall/pf.conf
     firewall/value_authorization astring solaris.smf.value.network.firewall
     ```

2. **Examine your firewall rules.**

   - **Examine the current rules in your firewall configuration.**

     The following example shows output from the packet filtering rule set that is loaded in the kernel.

     ```
     $ pfbash pfctl -s rules
     empty list for firewall(out)
     pass in quick on net1 from 198.51.100.0/27 to any flags S/SA
     block drop in on net1 all
     ```

   - **Verify that a possible firewall configuration is syntactically correct.**

     The following command checks the syntax of a rules file. This check does not load rules into the kernel.

     ```
     $ pfctl -n -f /test/firewall/pf.conf
     ```

3. **Examine the log files.**

   You can use utilities that read files in `libpcap` format, such as `tcpdump` or `tshark`. For more information, see the `tcpdump`(8), `tshark`(1), and `pcap`(3PCAP) man pages.

**Troubleshooting** If you expect log file entries but they are not in the log file, make sure that you have used a valid name for a capture datalink. Strings that match the following pattern are a safe choice for a capture datalink interface name: `[a-z][:alnum:]*[0-9]`. So, for example, dashes and underscores should not be used.

## ▼ How to Use Groups to Simplify Firewall Policy in a Network

Groups enable you to write one firewall policy for a type of network traffic, then apply the policy to the appropriate interfaces on different systems. Members of an interface group handle similar types of traffic. For example, a group of interfaces that filter LAN traffic could be added to the group "`lan`".

1. **Create a group name by using the name in a rule in the PF configuration file.**

   Typical names are `lan`, `wan`, `email`, and `imap`. The names should identify the purpose of the group, such as the type of traffic being filtered.

   .

   For example, the following rule uses the group name `lan`.

   ```
   pass in on lan from 198.51.100.0/27 to any
   ```

2. **On each system where the PF configuration file contains the group name, attach the group to the interface that performs that function.**

   PF cannot use the group name to filter packets until you make the group name a property of the interface.

   For example, on `HostB`, the `net1` and `net2` interfaces provide the LAN firewall. The following command makes the group a property of the interface.

   ```
   HostB# ipadm set-ifprop -p fwifgroup+=lan -m ip net1
   HostB# ipadm set-ifprop -p fwifgroup+=lan -m ip net2
   ```

   **Tip -** To modify the active interface group configuration for testing purposes, use the `-t` option to make the change temporary. Upon reboot, the initial interface group configuration is restored.

3. **Verify the group assignments.**

   See Example 10, “Showing, Testing, and Deleting Firewall Interface Groups,” on page 74. For a fuller example, see Example 11, “PF Configuration File Using Firewall Interface Groups,” on page 76.

**Example 10** Showing, Testing, and Deleting Firewall Interface Groups

In this example, the administrator uses the `-t` option to the `ipadm` command to modify the active PF configuration file in the preceding task on two hosts. On reboot, the original persistent configuration is restored.

`HostA` has the following interfaces: `net0`, `net1`, `vnic3`, `vnic4`, and `vnic5`.

`HostB` has the following interfaces: `net0`, `net1`, and `net2`.

1. The administrator adds the appropriate interfaces to the `lan` group.

```
HostA# for i in `ipadm show-if -p -o ifname|grep vnic` ; do
> ipadm set-ifprop -t -p fwifgroup+=lan -m ip $i;
> done

HostB# ipadm set-ifprop -t -p fwifgroup+=lan -m ip net1
HostB# ipadm set-ifprop -t -p fwifgroup+=lan -m ip net2
```

2. On each system, the administrator lists the firewall interface groups.

```
HostA$ ipadm show-ifprop -p fwifgroup
IFNAME      PROPERTY        PROTO PERM CURRENT    PERSISTENT DEFAULT    POSSIBLE
lo0         fwifgroup       ip    rw   --         --          --        --
net0        fwifgroup       ip    rw   --         --          --        --
net1        fwifgroup       ip    rw   --         --         --         --
vnic3       fwifgroup       ip    rw   lan        --         --         lan
vnic4       fwifgroup       ip    rw   lan        --         --         lan
vnic5       fwifgroup       ip    rw   lan        --         --         lan
HostB$ ipadm show-ifprop -p fwifgroup
IFNAME      PROPERTY        PROTO PERM CURRENT    PERSISTENT DEFAULT    POSSIBLE
lo0         fwifgroup       ip    rw   --         --          --        --
net0        fwifgroup       ip    rw   lan        --          --        lan
net1        fwifgroup       ip    rw   lan        --          --        lan
```

3. The administrator can also list the firewall interface groups per interface.

```
HostA$ ipadm show-ifprop -p fwifgroup vnic3
IFNAME      PROPERTY        PROTO PERM CURRENT    PERSISTENT DEFAULT    POSSIBLE
vnic3       fwifgroup       ip    rw   lan        --         --         lan
```

4. After some testing, the administrator removes `net0` from the `lan` firewall interface group on `HostB`, and removes all interface groups from `vnic3` on `HostA`.

```
HostB# ipadm set-ifprop -p fwifgroup-=lan -m ip net0
HostA$ ipadm reset-ifprop -t -p fwifgroup -m ip vnic3
```

- The `fwifgroup-` argument removes one group assignment from `net0`.

```
HostB# ipadm set-ifprop -p fwifgroup-=lan -m ip net0
```

- The `reset-ifprop` subcommand removes all group assignments from `vnic3`.

```
HostA# ipadm reset-ifprop -t -p fwifgroup -m ip vnic3
```

If the administrator removes all firewall interface group assignments, the group name remains in the PF configuration file but is not used for packet filtering.

## ▼ How to Remove Packages That Are Dependent on the `firewall` Package

If you do not plan to use PF to do NAT, or to use the `ftp-proxy` service, or to log PF packets, you can remove the packages that install these PF functions.

**Before You Begin** To remove packages, you must become an administrator who is assigned the Software Installation rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Uninstall the FTP proxy package.**

   By uninstalling this package, you cannot use PF to do NAT.

   ```
   $ pfexec pkg uninstall firewall/firewall-ftp-proxy
   ```

2. **Uninstall the `firewall-pflog` package.**

   By uninstalling this package, you cannot log packets that go through the firewall.

   ```
   $ pfexec pkg uninstall firewall/firewall-pflog
   ```

## Examples of PF Configuration Files

The examples in this section illustrate PF rules and rule sets.

Configuration files follow standard UNIX syntax rules:

- The pound sign (`#`) indicates a comment.
- Rules and comments can coexist on the same line.
- Extraneous white space is allowed for readability.

- Rules can be more than one line long. A backslash (\) at the end of a line indicates that the rule continues on the next line.

For more detailed syntax information, see "Packet Filter Rule Syntax" on page 48 and the `pf.conf(7)` man page.

**EXAMPLE 11** PF Configuration File Using Firewall Interface Groups

This example shows a configuration on two host systems with multiple network interfaces. This example extends the example used in Example 10, "Showing, Testing, and Deleting Firewall Interface Groups," on page 74.

The same PF configuration file is loaded on both systems, but the `lan`, `wan`, and `imap` firewall interface groups have different members on the two hosts. The administrator sets the `fwifgroup` property with the `ipadm` command per interface to add the interfaces to the groups that enforce the policy.

`HostA` has the following interfaces: `net0`, `net1`, `vnic3`, `vnic4`, and `vnic5`.

`HostB` has the following interfaces: `net0`, `net1`, and `net2`.

The following PF configuration file is loaded at boot on both systems:

```
## Policy for members of lan, imap, and wan groups
        block from any to any
        pass in on lan from 198.51.100.0/27 to any
        pass quick on imap proto tcp from 198.51.100.0/27 to any \
            port=imaps received-on lan
        pass out on wan from 198.51.100.0/27 to any received-on lan
...
```

The following commands ensure that the interfaces are added to the correct groups. On `HostA`, the administrator runs the following commands:

```
# ipadm set-ifprop -p fwifgroup+=wan -m ip net0
# ipadm set-ifprop -p fwifgroup+=imap -m ip net1
# for i in `ipadm show-if -p -o ifname|grep vnic` ; do
> ipadm set-ifprop -p fwifgroup+=lan -m ip $i;
> done
```

On `HostB`, the administrator runs the following commands:

```
# ipadm set-ifprop -p fwifgroup+=wan -m ip net0
# ipadm set-ifprop -p fwifgroup+=imap -m ip net0
# ipadm set-ifprop -p fwifgroup+=lan -m ip net1
# ipadm set-ifprop -p fwifgroup+=lan -m ip net2
```

**EXAMPLE 12** Sample PF Configuration File

This annotated file expands on the basic protection rule set. The firewall uses these rules whenever the `firewall` service goes into maintenance mode. For example, a syntax error might put the service into maintenance mode.

```
##  make IP reassembly work
set reassemble yes no-df

## block everything unless told otherwise
## and send TCP-RST/ICMP unreachable
## for every packet which gets blocked
block return

## accept incoming SSH connections
pass in proto tcp to any port 22

## allow incoming messages from DHCP
pass in inet proto udp from port 67 to port 68
pass in inet6 proto udp from port 547 to port 546

## packet too big - needed for PMTUD
pass in inet6 proto ipv6-icmp icmp6-type 2

## router advertisement
pass in inet6 proto ipv6-icmp icmp6-type 134

## neighbor solicitation
pass in inet6 proto ipv6-icmp icmp6-type 135

## neighbor advertisement
pass in inet6 proto ipv6-icmp icmp6-type 136

## allow all connections initiated from this system,
## including DHCP requests
pass out
```

## Examples of Converting IPF Rules to PF Rules

The following examples are linked, and show more complex conversions than the examples in "IP Filter to Packet Filter Rules Conversion Tool" on page 62. Several examples illustrate how you can optimize the PF rule sets that `ipf2pf` generates.

The examples cover the following scenarios:

- Example 13, "Blocking All IPv6 Traffic in PF," on page 78

- Example 14, "Converting IPF IPv4 and IPv6 Policies That Are Identical to PF," on page 79
- Example 15, "Removing `inet` and `inet6` Address Families From `ipf2pf` Output," on page 80
- Example 16, "Converting IPF NAT Rules to PF Rules," on page 81
- Example 17, "Optimizing Converted IPF NAT Rules in PF," on page 83
- Example 18, "Implementing the Correct NAT Policy," on page 84

**EXAMPLE 13** Blocking All IPv6 Traffic in PF

If your host does not handle IPv6 traffic, then it is good practice to block the IPv6 protocol at the firewall.

If you have IPv6 rules in the `/etc/ipf/ipf6.conf` file to block everything, as in the following `/etc/ipf/ipf6.conf` file, then you run the `ipf2pf` command on both files to convert them.

```
    ## IP Filter Sample ipf6.conf File
    #
    # block IPv6 completely
    #
    block in all
    block out all
```

```
$ ipf2pf -4 ipf.conf -6 ipf6.conf
```

```
    #
    # File was generated by ipf2pf(7) service during system upgrade. The service
    # attempted to convert your IPF rules to PF (the new firewall) rules. You
    # should check if firewall configuration here, suggested by ipf2pf, still meets
    # your network policy requirements.
    #

    #
    # Unlike IPF, PF intercepts packets on loopback by default.
    # IPF does not intercept packets bound to loopback. To turn off the
    # policy check for loopback packets, we suggest to use command
    # below:
    set skip on lo0
    #
    # PF does IP reassembly by default. It looks like your IPF
    # does not have IP reassembly enabled. Therefore the feature is
    # turned off.
    #
    set reassemble no
    # In case you change your mind and decide to enable IP reassembly
    # delete the line above. Also to improve interoperability
```

```
# with broken IP stacks, tell PF to ignore the 'DF' flag when
# doing reassembly. Uncommenting line below will do it:
#

# set reassemble yes no-df

block drop in inet all
#
# IPF rule specifies either a port match or return-rst action,
# but does not specify a protocol (TCP or UDP). PF requires a port
# rule to include a protocol match using the 'proto' keyword.
# ipf2pf always assumes and enters a TCP port number
#

pass in inet proto tcp from any to any port = 22 flags any keep state (sloppy)
pass out inet all flags any keep state (sloppy)
block drop in inet6 all
block drop out inet6 all
```

**Note -** The `ipf2pf` conversion deliberately relaxes stateful policy to `sloppy` by specifying a `keep state (sloppy)` rule that is not visible in the file. This policy keeps firewall behavior close to IPF behavior.

If your policy requires a strict state inspection for TCP, you can change it to `keep state on lo0`. This might be your first customization of the `ipf2pf` conversion.

**EXAMPLE 14** Converting IPF IPv4 and IPv6 Policies That Are Identical to PF

If your network policy for IPv6 is to be same as for IPv4, that is, block every remote packet coming in except `ssh`, and allow everything out. The following output is the result of conversion of those rules:

```
#
# File was generated by ipf2pf(7) service during system upgrade. The service
# attempted to convert your IPF rules to PF (the new firewall) rules. You
# should check if firewall configuration here, suggested by ipf2pf, still meets
# your network policy requirements.
#

#
# Unlike IPF, PF intercepts packets on loopback by default.
# IPF does not intercept packets bound to loopback. To turn off the
# policy check for loopback packets, we suggest to use command
# below:
set skip on lo0
#
```

```
# PF does IP reassembly by default. It looks like your IPF
# does not have IP reassembly enabled. Therefore the feature is
# turned off.
#
set reassemble no
# In case you change your mind and decide to enable IP reassembly
# delete the line above. Also to improve interoperability
# with broken IP stacks, tell PF to ignore the 'DF' flag when
# doing reassembly. Uncommenting line below will do it:
#

# set reassemble yes no-df

block drop in inet all
#
# IPF rule specifies either a port match or return-rst action,
# but does not specify a protocol (TCP or UDP). PF requires a port
# rule to include a protocol match using the 'proto' keyword.
# ipf2pf always assumes and enters a TCP port number
#

pass in inet proto tcp from any to any port = 22 flags any keep state (sloppy)
pass out inet all flags any keep state (sloppy)
block drop in inet6 all
#
# IPF rule specifies either a port match or return-rst action,
# but does not specify a protocol (TCP or UDP). PF requires a port
# rule to include a protocol match using the 'proto' keyword.
# ipf2pf always assumes and enters a TCP port number
#

pass in inet6 proto tcp from any to any port = 22 flags any keep state (sloppy)
pass out inet6 all flags any keep state (sloppy)
```

The output of the `ipf2pf` command creates a rules file that duplicates IPv4 and IPv6 policy like IPF does. Because PF treats IPv4 and IPv6 rules in one rule set, you could manually optimize the `ipf2pf` rule set file to remove the address family (`inet` and `inet6`) at `pass` and `block` rules, as shown in the following example.

**EXAMPLE 15** Removing `inet` and `inet6` Address Families From `ipf2pf` Output

```
## Streamlined ipf2pf output to remove inet/d references
# File was generated by ipf2pf(7) service during system upgrade. The service
# attempted to convert your IPF rules to PF (the new firewall) rules. You
# should check if firewall configuration here, suggested by ipf2pf, still meets
# your network policy requirements.
#
```

```
#
# Unlike IPF, PF intercepts packets on loopback by default.
# IPF does not intercept packets bound to loopback. To turn off the
# policy check for loopback packets, we suggest to use command
# below:
set skip on lo0
#
# PF does IP reassembly by default. It looks like your IPF
# does not have IP reassembly enabled. Therefore the feature is
# turned off.
#
set reassemble no
# In case you change your mind and decide to enable IP reassembly
# delete the line above. Also to improve interoperability
# with broken IP stacks, tell PF to ignore the 'DF' flag when
# doing reassembly. Uncommenting line below will do it:
#

# set reassemble yes no-df

block drop in inet all
#
# IPF rule specifies either a port match or return-rst action,
# but does not specify a protocol (TCP or UDP). PF requires a port
# rule to include a protocol match using the 'proto' keyword.
# ipf2pf always assumes and enters a TCP port number
#

pass in proto tcp from any to any port = 22 flags any keep state (sloppy)
pass out all flags any keep state (sloppy)
```

**EXAMPLE 16** Converting IPF NAT Rules to PF Rules

Unlike PF, IPF also uses separate rules for NAT from the `/etc/ipf/ipnat.conf` file. The following shows a typical `ipnat.conf` configuration:

```
## IP NAT Configuration File, ipnat.conf
# Translate SRC address of LAN to IP address bound to net0
#
map net0 from 198.51.100.0/24 to any -> 0.0.0.0
#
# redirect ssh sessions from internet to SSH gate (198.51.100.22)
#
rdr net0 from any to 203.0.113.15 port = 22 -> 198.51.100.22 port 22
```

To add the preceding NAT PF policy to you IPv4 and IPv6 IPF policy, run the following command:

```
$ ipf2pf -4 ipf.conf -6 ipf6.conf -n ipnat.conf
```

This command generates the following output when you add the NAT rules if your network policy:

- Blocks inbound IPv6 completely
- Allows IPv4 `ssh` inbound sessions only
- Allows unrestricted outbound IPv6 traffic

```
#
# File was generated by ipf2pf(7) service during system upgrade. The service
# attempted to convert your IPF rules to PF (the new firewall) rules. You
# should check if firewall configuration here, suggested by ipf2pf, still meets
# your network policy requirements.
#

#
# Unlike IPF, PF intercepts packets on loopback by default.
# IPF does not intercept packets bound to loopback. To turn off the
# policy check for loopback packets, we suggest to use command
# below:
set skip on lo0
#
# PF does IP reassembly by default. It looks like your IPF
# does not have IP reassembly enabled. Therefore the feature is
# turned off.
#
set reassemble no
# In case you change your mind and decide to enable IP reassembly
# delete the line above. Also to improve interoperability
# with broken IP stacks, tell PF to ignore the 'DF' flag when
# doing reassembly. Uncommenting line below will do it:
#

# set reassemble yes no-df

#
# Unlike IPF, the PF firewall implements NAT as yet another
# optional action of a regular policy rule. To keep PF
# configuration close to the original IPF, consider using
# the 'match' action in PF rules, which translate addresses.
# There is one caveat with 'match'. You must always write a 'pass'
# rule to match the translated packet. Packets are not translated
# unless they hit a subsequent pass rule. Otherwise, the "match"
# rule has no effect.
```

```
   #

   # It's also important to avoid applying nat rules to DHCP/BOOTP
   # requests. The following stateful rule, when added above the NAT
   # rules, will avoid that for us.
   pass out quick proto udp from 0.0.0.0/32 port 68 to 255.255.255.255/32 port 67

   # There are 2 such rules in your IPF ruleset
   #
   match out on net0 inet from 198.51.100.0/24 to any nat-to (net0)
   match in on net0 inet proto tcp from any to 203.0.113.15 port = 22 rdr-to
198.51.100.22 port 22
   match in on net0 inet proto udp from any to 203.0.113.15 port = 22 rdr-to
198.51.100.22 port 22


   #
   # The pass rules below make sure rdr/nat -to actions
   # in the match rules above will take effect.
   pass out all
   pass in all
   block drop in inet all
   #
   # IPF rule specifies either a port match or return-rst action,
   # but does not specify a protocol (TCP or UDP). PF requires a port
   # rule to include a protocol match using the 'proto' keyword.
   # ipf2pf always assumes and enters a TCP port number
   #

   pass in inet proto tcp from any to any port = 22 flags any keep state (sloppy)
   pass out inet all flags any keep state (sloppy)
   block drop in inet6 all
   block drop out inet6 all
```

**EXAMPLE 17** Optimizing Converted IPF NAT Rules in PF

NAT rules that `ipf2pf` generates can be optimized. For example, if the IP address of the `net0` interface (`203.0.113.15`) is statically assigned, then the '`pass out quick...`' rule, which deals with DHCP, is unnecessary. You can also merge `rdr-to` actions in `match` rules to `pass in` rules to clarify the rule set further, as shown in the following optimization of the NAT rules in Example 16, “Converting IPF NAT Rules to PF Rules,” on page 81:

```
. . .
   #
   # Unlike IPF, the PF firewall implements NAT as yet another
   # optional action of a regular policy rule. To keep PF
   # configuration close to the original IPF, consider using
```

```
    # the 'match' action in PF rules, which translate addresses.
    # There is one caveat with 'match'. You must always write a 'pass'
    # rule to match the translated packet. Packets are not translated
    # unless they hit a subsequent pass rule. Otherwise, the "match"
    # rule has no effect.
    #

    match out on net0 inet from 198.51.100.0/24 to any nat-to (net0)
    pass out all
    pass in all
    block drop in inet all
    pass out inet all flags any keep state (sloppy)
    pass in on net0 inet proto tcp from any to (net0) port = 22 rdr-to 198.51.100.22
 port 22
    block drop in inet6 all
    block drop out inet6 all
```

**EXAMPLE 18** Implementing the Correct NAT Policy

The remaining match rule with the `nat-to` action can be merged to a '`pass out`' rule, too. However, before merging it, consider that the `pass out` rule matches two sets of packets: packets that are forwarded for all LAN hosts in the `198.51.100.0/24` network, and local outbound packets from the firewall host itself, that is, local outbound traffic.

A change in the NAT rule can implement a different network policy.

1. If the firewall host is not supposed to talk to the public network, that is, to the network attached to `net0`, then you can merge the `nat-to` rule to the `pass` rule as you did for `rdr-to`:

   ```
   pass out on net0 inet from 198.51.100.0/24 to any nat-to (net0)
   ```
2. If the firewall host must talk to the network, then you can optimize the rule, although the conversion rule is sufficient:

   ```
   pass out on net0 inet from any to any nat-to (net0)
   ```

   This rule also allows the firewall host to talk to the Internet. Also note that the `nat-to` action is not applied to packets with the same address as the address that is bound to `net0`. For those packets, NAT translation is not applied. Also keep in mind that the preceding rule matches all outbound packets at `net0` NIC, including packets which might be coming from networks other than the `198.51.100.0/24` network.

♦ ♦ ♦ CHAPTER 6

# About IP Security Architecture

IP Security Architecture (IPsec) provides cryptographic protection for IP packets in IPv4 and IPv6 networks.

This chapter covers the following topics:

- "Introduction to IPsec" on page 85
- "IPsec Packet Flow" on page 87
- "IPsec Security Associations" on page 90
- "IPsec Protection Protocols" on page 91
- "IPsec Policy" on page 94
- "Transport and Tunnel Modes in IPsec" on page 95
- "Virtual Private Networks and IPsec" on page 97
- "IPsec and FIPS 140-2" on page 98
- "IPsec and NAT Traversal" on page 99
- "IPsec and SCTP" on page 100
- "IPsec and Oracle Solaris Zones" on page 100
- "IPsec and Virtual Machines" on page 101
- "IPsec Configuration Commands and Files" on page 101

To implement IPsec on your network, see Chapter 7, "Configuring IPsec". For reference information, see Chapter 12, "IPsec and Key Management Reference".

## Introduction to IPsec

IPsec protects the contents of IP packets by using encryption and provides integrity checking by authenticating the packet contents. Because IPsec is performed at the network layer, a network application can take advantage of IPsec while not having to configure itself to use IPsec. When used properly, IPsec is an effective tool in securing network traffic.

IPsec uses the following terms:

- **Security protocols –** The protection that is applied to an IP packet. The authentication header (AH) protects a IP packet by adding an integrity check vector (ICV) which is a hash of the complete packet including the IP headers. The receiver is assured that the packet has not been modified. It does not provide confidentiality with encryption.

  The encapsulating security payload (ESP) protects the payload of an IP packet. The payload of a packet can be encrypted to provide confidentiality and can ensure data integrity by using an ICV.
- **Security associations (SA) –** The cryptographic parameters, keys, IP security protocol, IP addresses, IP protocol, port numbers, and other parameters that are used to match a particular SA to a specific traffic flow.
- **Security associations database (SADB) –** The database that stores the security associations. SAs are referenced by the security parameter index (SPI), security protocol, and destination IP address. These three elements uniquely identify an IPsec SA. When a system receives an IP packet which has an IPsec header (ESP or AH), the system searches the SADB for a matching SA. If a matching SA is found, it is used to allow IPsec to decrypt and verify the packet. If verification fails or no matching SA is found, the packet is discarded.
- **Key management –** The secure generation and distribution of keys that are used by cryptographic algorithms and the generation of the SAs used to store them.
- **Security policy database (SPD) –** The database that specifies the security policy to apply to IP traffic. The SPD filters the traffic to determine how the packets should be processed. A packet can be discarded or passed in the clear. Or, a packet can be protected with IPsec, that is, the security policy is applied.

  For outbound packets, the IPsec policy determines whether IPsec should be applied to an IP packet. If IPsec is applied, the IP module searches the SADB for a matching SA and uses this SA to enforce the policy.

  For inbound packets, the IPsec policy ensures that the protection level of a received packet is appropriate. If the policy requires packets from a certain IP address to be protected by IPsec, the system discards any unprotected packets. If an inbound packet is protected by IPsec, the IP module searches the SADB for a matching SA and applies the SA to the packet.

Applications can invoke IPsec to apply security mechanisms to IP packets on a per-socket level as well. If a socket on a port is connected and IPsec policy is later applied to that port, then traffic that uses that socket is not protected by IPsec. Of course, a socket that is opened on a port *after* IPsec policy is applied to the port is protected by IPsec policy.

## IPsec Packet Flow

Figure 3, “IPsec Applied to Outbound Packet Process,” on page 88 shows how an IP packet proceeds when IPsec has been invoked on an outbound packet. The flow diagram illustrates where authentication header (AH) and encapsulating security payload (ESP) entities can be applied to the packet. Subsequent sections describe how to apply these entities, as well as how to choose the algorithms.

Figure 4, “IPsec Applied to Inbound Packet Process,” on page 89 shows the IPsec inbound process.

**FIGURE 3** IPsec Applied to Outbound Packet Process

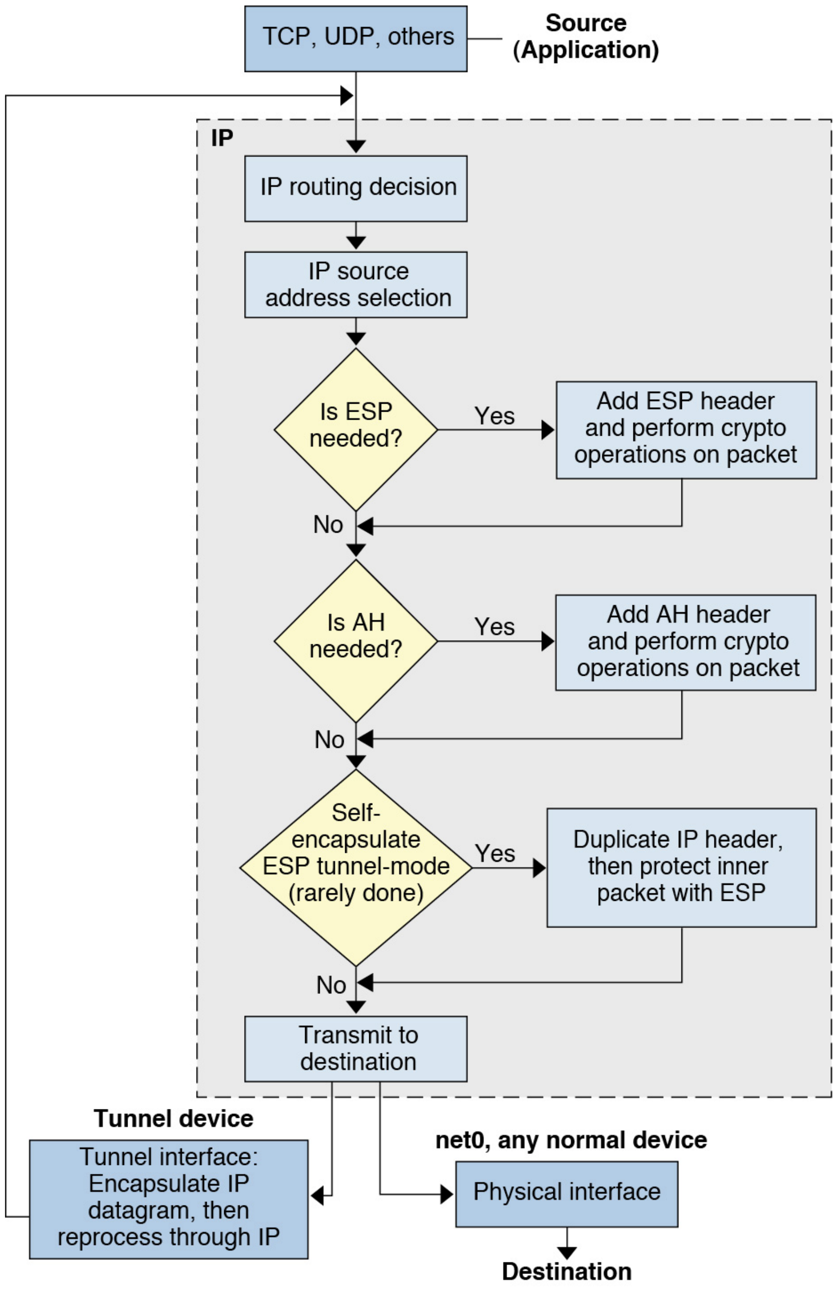
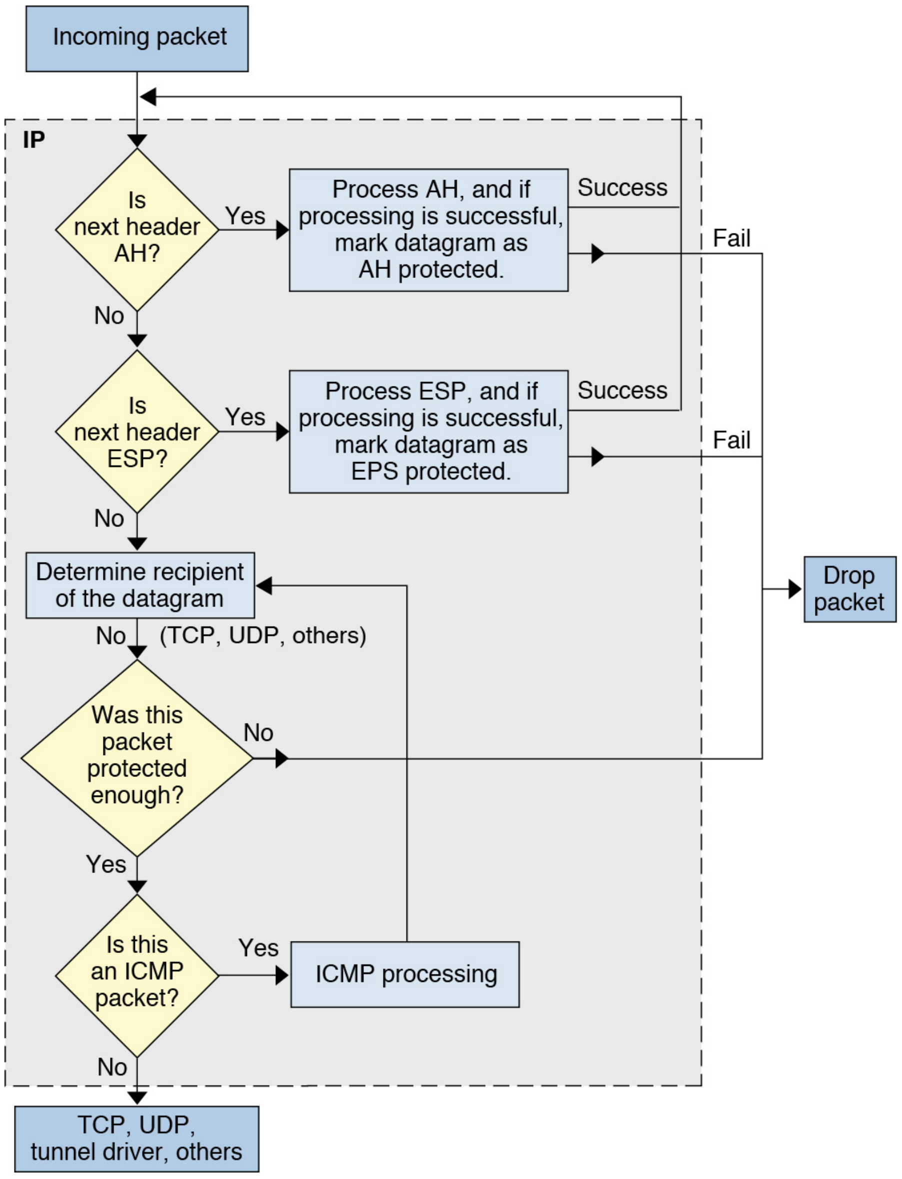

TCP, UDP, others
Source (Application)
IP
IP routing decision
IP source address selection
Is ESP needed?
Yes
Add ESP header and perform crypto operations on packet
No
Is AH needed?
Yes
Add AH header and perform crypto operations on packet
No
Self-encapsulate ESP tunnel-mode (rarely done)
Yes
Duplicate IP header, then protect inner packet with ESP
No
Transmit to destination
Tunnel device
Tunnel interface: Encapsulate IP datagram, then reprocess through IP
net0, any normal device
Physical interface
Destination

**FIGURE 4** IPsec Applied to Inbound Packet Process


Incoming packet
IP
Is next header AH?
Yes
Process AH, and if processing is successful, mark datagram as AH protected.
Success
Fail
No
Is next header ESP?
Yes
Process ESP, and if processing is successful, mark datagram as EPS protected.
Success
Fail
No
Determine recipient of the datagram
Drop packet
No
(TCP, UDP, others)
Was this packet protected enough?
No
Yes
Is this an ICMP packet?
Yes
ICMP processing
No
TCP, UDP, tunnel driver, others

## IPsec Security Associations

An IPsec *security association* SA defines the security properties that will be applied to an IP packet that matches the IP parameters that are also stored in the SA. Each SA is unidirectional. Because most communications are bidirectional, two SAs are required for a single connection.

Together, the following three elements uniquely identify an IPsec SA:

- The security protocol (AH or ESP)
- The destination IP address
- The security parameter index (SPI)

The SPI of the SA provides additional protection and is transmitted in the AH or ESP header of an IPsec-protected packet. The `ipsecah`(4P) and `ipsecesp`(4P) man pages explain the extent of protection that is provided by AH and ESP. An integrity checksum value is used to authenticate a packet. If the authentication fails, the packet is dropped.

Security associations are stored in a *security associations database* (SADB). A socket-based administrative interface, `PF_KEY` enables privileged applications to manage the database programmatically. For example, the IKE daemon and the `ipseckey` command use the `PF_KEY` socket interface.

For a more complete description of the IPsec SADB, see "Security Associations Database for IPsec" on page 222.

For more information about how to manage the SADB, see the `pf_key`(4P) and `ipseckey`(8) man pages.

## Key Management for IPsec Security Associations

Security associations (SAs) require keying material for authentication and for encryption. The managing of this keying material is called *key management*. Oracle Solaris provides two methods for managing the keys for IPsec SAs: IKE and manual key management.

### IKE for IPsec SA Generation

The Internet Key Exchange (IKE) protocol handles key management automatically. This Oracle Solaris release supports IKE version 2 (IKEv2) and IKE version 1 (IKEv1) of the IKE protocol.

The use of IKE to manage IPsec SAs is encouraged. These key management protocols offer the following advantages:

- Simple configuration
- Provide strong peer authentication
- Automatically generate SAs with a high quality random key source
- Do not require administrative intervention to generate new SAs

For more information, see "How IKE Works" on page 131.

To configure IKE, see Chapter 9, "Configuring IKEv2". If you are communicating with a system that does not support the IKEv2 protocol, follow the instructions in Chapter 10, "Configuring IKEv1".

### Manual Keys for IPsec SA Generation

The use of manual keys is more complicated than IKE and is potentially risky. A system file, `/etc/inet/secret/ipseckeys`, contains the encryption keys. If these keys are compromised, they can be used to decrypt recorded network traffic. Because IKE frequently changes the keys, the window of exposure to such a compromise is much smaller. Using the `ipseckeys` file or its command interface, `ipseckey`, is appropriate only for systems that do not support IKE.

While the `ipseckey` command has only a limited number of general options, the command supports a rich command language. You can specify that requests be delivered by means of a programmatic interface specific for manual keying. For additional information, see the `ipseckey`(8) and `pf_key`(4P) man pages.

Typically, manual SA generation is used when IKE is unavailable for some reason. However, if the SPI values are unique, manual SA generation and IKE can be used at the same time.

## IPsec Protection Protocols

IPsec provides two security protocols for protecting data:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

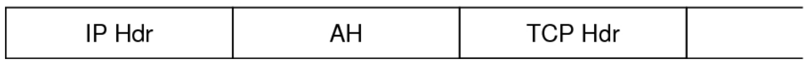
AH provides data integrity by using an authentication algorithm. It does not encrypt the packet.

ESP typically protects the packet with an encryption algorithm and provides data integrity with an authentication algorithm. Some encryption algorithms provide both encryption and authentication, such as AES GCM.

The AH protocol cannot be used with network address translation (NAT).

## Authentication Header

The authentication header provides data authentication, strong integrity, and replay protection to IP packets. AH protects the greater part of the IP packet. As the following illustration shows, AH is inserted between the IP header and the transport header.

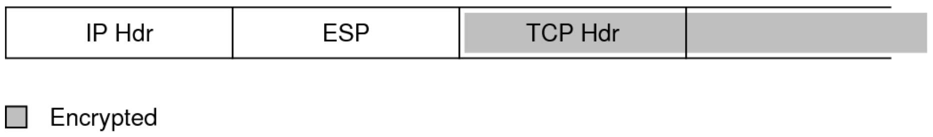


The transport header can be TCP, UDP, SCTP, or ICMP. If a tunnel is being used, the transport header can be another IP header.

## Encapsulating Security Payload

The encapsulating security payload (ESP) protocol provides confidentiality over what the ESP encapsulates. ESP also provides the services that AH provides. However, ESP does not protect the outer IP header. ESP provides authentication services to ensure the integrity of the protected packet. Because ESP uses encryption-enabling technology, a system that provides ESP can be subject to import and export control laws.

The ESP header and trailer encapsulate the IP payload. When encryption is used with ESP, it is applied only over the IP payload data, as shown in the following illustration.



In a TCP packet, the ESP header is authenticated and it encapsulates the TCP header and its data. If the packet is an IP-in-IP packet, ESP protects the inner IP packet. Per-socket policy allows *self-encapsulation*, so ESP can encapsulate IP options when necessary.

Self-encapsulation can be used by writing a program that uses the `setsockopt()` system call. If self-encapsulation is set, a copy of the IP header is made to construct an IP-in-IP packet. For example, when self-encapsulation is not set on a TCP socket, the packet is sent in the following format:

```
[ IP(a -> b) options + TCP + data ]
```

When self-encapsulation is set on that TCP socket, the packet is sent in the following format:

```
[ IP(a -> b) + ESP [ IP(a -> b) options + TCP + data ] ]
```

For further discussion, see "Transport and Tunnel Modes in IPsec" on page 95.

## Security Considerations When Using AH and ESP

The following table compares the protections that are provided by AH and ESP.

**TABLE 5** Protections Provided by AH and ESP in IPsec

| Protocol | Packet Coverage | Protection | Against Attacks |
|---|---|---|---|
| AH | Protects packet from the IP header to the end of the transport data | Provides strong integrity, data authentication:<br>■ Ensures that the receiver receives exactly what the sender sent<br>■ Is susceptible to replay attacks when an AH does not enable replay protection | Replay, cut-and-paste |
| ESP | Protects packet from the ESP header to the end of the transport data | With encryption option, encrypts the IP payload. Ensures confidentiality | Eavesdropping |
| | | With authentication option, provides the same payload protection as AH | Replay, cut-and-paste |
| | | With both options, provides strong integrity, data authentication, and confidentiality | Replay, cut-and-paste, eavesdropping |

## Authentication and Encryption Algorithms in IPsec

IPsec security uses two types of algorithms, authentication and encryption. The AH protocol uses authentication algorithms. The ESP protocol can use encryption as well as authentication algorithms. You can obtain a list of the algorithms on your system and their properties by using the `ipsecalgs` command. For more information, see the `ipsecalgs`(8) man page. You can also use the functions that are described in the `getipsecalgbyname`(3C) man page to retrieve the properties of algorithms. IPsec uses the Cryptographic Framework to perform encryption and authentication.

For more information, see the following:

- Chapter 1, "About Cryptographic Providers in Oracle Solaris" in *Managing Encryption and Certificates in Oracle Solaris 11.4*
- Chapter 7, "Introduction to the Oracle Solaris Cryptographic Framework" in *Developer's Guide to Oracle Solaris 11.4 Security*

## IPsec Policy

IPsec policy can be applied at the per-socket level and the system-wide level.

IPsec applies policy to outbound packets and inbound packets that match an IPsec policy rule. Each policy rule can have one or more actions. An action could be to encrypt a packet by using a specific algorithm, or to pass the packet without encryption. To specify multiple acceptable algorithms, a policy rule would use multiple actions.

An IPsec policy rule has three parts.

- Selectors – Determine if a rule matches the network packet. Also known as traffic selectors. An empty (`{}`) selector matches all traffic. A selector can have more than one action/ parameter pair.
- Action – Applied when traffic matches the selectors. Examples of actions include `ipsec` and `pass`.
- Action parameters – Additional specifications for an action. Simple actions like `pass` or `drop` do not have parameters. Actions such as `ipsec` can specify cryptographic parameters.

IPsec policy is applied to both inbound and outbound packets. A packet that does not match any rule is passed. When packets can match more than one rule, the first match is used.

The rules are processed in the following order:

1. Per-socket rules
2. System-wide `pass`, `bypass`, and `drop` rules
3. System-wide `ipsec` rules that use ESP
4. System-wide `ipsec` rules that use AH

The `bypass` and `or pass` options specify exceptions to an IPsec policy rule that otherwise applies to the packets.

- You can `bypass` all or part of an IPsec rule. Packets matching a bypass rule will be allowed to pass without IPsec protection and any other IPsec policy rules that match the packets are not applied. For example, packets from web clients might not need to be encrypted. See "How to Use IPsec to Protect Web Server Communication With Other Servers" on page 109.
- The `or pass` action in an IPsec policy rule enables non-IPsec packets that match a previous action in the rule to pass into the system. An IPsec policy rule which has an `encrypt` action and an `or pass` action accepts encrypted packets and packets that are not encrypted from the client systems.

  The `or pass` action enables a server to serve clients that are not configured with IPsec as well as clients that are. One example of use would be when a network is in transition to configuring IPsec on every system. This option is not suitable for an environment where all traffic must be encrypted. For an example, see Example 22, "Transitioning Client Systems to Use IPsec by Using the `or pass` Action on the Server," on page 109.

You use the `ipsecinit.conf` file and the `ipsecconf` command to configure IPsec policy. For details and examples, see the `ipsecconf(8)` man page and Chapter 7, "Configuring IPsec".

## Transport and Tunnel Modes in IPsec

The IPsec standards define two distinct modes of IPsec operation, *transport mode* and *tunnel mode*. The key difference between transport and tunnel mode is where policy is applied. In tunnel mode, the original packet is encapsulated in an outer IP header. The IP addresses in the inner and outer headers can be different.

Traffic selectors, introduced in "IPsec Policy" on page 94, determine if a packet matches a policy rule. Selectors include:

- Source IP address
- Destination IP address
- Protocol number, if applicable
- Port numbers, if applicable

The pattern used to match IPsec policy rules consists of a subset of these selectors.

In transport mode, the traffic selectors are matched against the outer IP header. In tunnel mode, they are matched against the inner IP header. Tunnel mode can be applied to any mix of end systems and intermediate systems, such as security gateways.

In transport mode, the IP header, the next header, and any ports that the next header supports can be used to determine if IPsec policy applies. In effect, IPsec can enforce different transport mode policies between two IP addresses to the granularity of a single port. For example, if the next header is TCP, which supports ports, then IPsec policy can be set for a TCP port of the outer IP address.

Tunnel mode works only for IP-in-IP packets. In tunnel mode, IPsec policy is enforced on the contents of the inner IP packet. Different policy can be enforced for different inner IP addresses. That is, the inner IP header, its next header, and the ports that the next header supports can enforce a policy.

In tunnel mode, IPsec policy can be specified for subnets of a LAN behind a router and for ports on those subnets. IPsec policy can also be specified for particular IP addresses, that is, hosts, on those subnets. The ports of those hosts can also have a specific IPsec policy. For examples of tunneling procedures that include configuring static routes, see "Protecting a VPN With IPsec" on page 112.

When IPsec policy is applied to traffic in IP tunnels, the name of the IP tunnel interface is used to link the traffic in that tunnel to an IPsec policy rule. IPsec policy provides a `tunnel` keyword to select an IP tunneling network interface. When the `tunnel` keyword is present in a rule, all selectors that are specified in that rule apply to the inner packet.

For information about tunneling interfaces, see Chapter 4, "About IP Tunnel Administration" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

The following figures illustrate protected and unprotected packets.

Figure 5, "Unprotected IP Packet Carrying TCP Information," on page 96 shows an IP header with an unprotected TCP packet.

**FIGURE 5** Unprotected IP Packet Carrying TCP Information

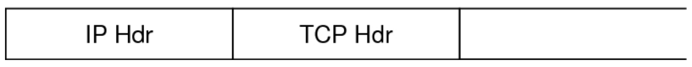


Figure 6, "Protected IP Packet Carrying TCP Information," on page 97 shows ESP protecting the data in transport mode. The shaded area shows the encrypted part of the packet.

**FIGURE 6** Protected IP Packet Carrying TCP Information

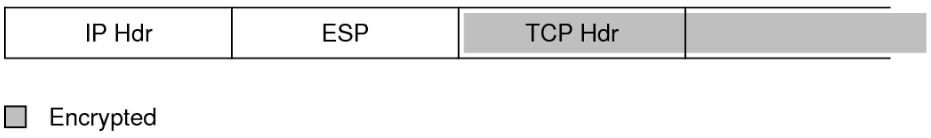
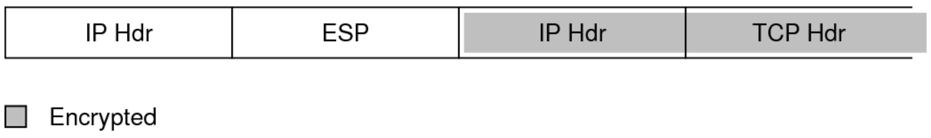


Figure 7, "IPsec Packet Protected in Tunnel Mode," on page 97 shows that the entire packet is *inside* the ESP header in tunnel mode. The packet from Figure 5, "Unprotected IP Packet Carrying TCP Information," on page 96 is protected in tunnel mode by an outer IPsec header and, in this case, ESP.

**FIGURE 7** IPsec Packet Protected in Tunnel Mode



IPsec policy provides keywords for tunnel mode and transport mode. For more information, review the following:

- For details on per-socket policy, see the `ipsec`(4P) man page.
- For an example of per-socket policy, see "How to Use IPsec to Protect Web Server Communication With Other Servers" on page 109.
- For more information about tunnels, see the `ipsecconf`(8) man page.
- For an example of tunnel configuration, see "How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode" on page 116.

## Virtual Private Networks and IPsec

The term private network (VPN) is often used to describe a private, secure, point-to-point network that is built over a more public network, for example, the Internet. The point-to-point network, or VPN, can be used to connect systems on private networks, or networks of systems on private networks together.

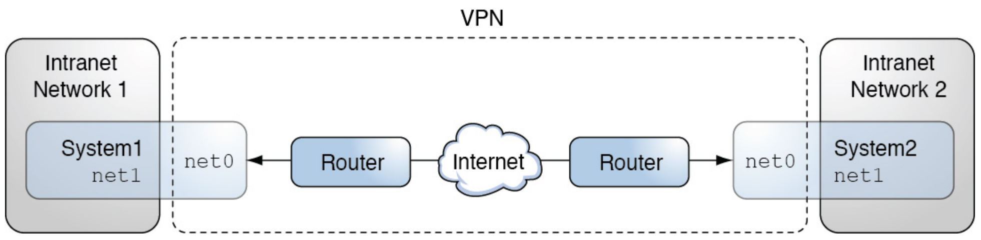
A configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see "How to Create and Configure an IP Tunnel" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

A tunnel creates an apparent physical interface to IP. IP traffic that passes over the IP tunnel interface can be protected with IPsec.

The tunnel interface in Oracle Solaris can be used to encapsulate, or *tunnel*, an IP packet from one system to another system. The tunneled packet adds an IP header in front of the original IP header. The added header uses addresses that are routable on the public network. These addresses are represented by the `net0` interfaces in the following diagram.

The following figure illustrates how two sites can use IPsec to create a VPN between them. Traffic between `Intranet 1` and `Intranet 2` is tunneled over the Internet by using IP-in-ESP encapsulation. In this case, the `net0` addresses are used in the outer IP headers, while the inner IP addresses are those of the tunneled packets from the intranet networks. Because the inner IP addresses are covered by ESP, they are protected from inspection as traffic crosses the Internet.

**FIGURE 8** Virtual Private Network



For a detailed example of the setup procedure, see "How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode" on page 116.

## IPsec and FIPS 140-2

On a FIPS 140-2 enabled system, you are responsible for choosing only FIPS 140-2 approved algorithms when creating certificates and configuring IPsec. The procedures and examples in this guide use FIPS 140-2 approved algorithms except when the algorithm "`any`" is specified.

**Note -** If you have a strict requirement to use only FIPS 140-2 validated cryptography, you must be running the Oracle Solaris 11.3 SRU 5.6 release. Oracle completed a FIPS 140-2 validation against the Cryptographic Framework in this specific release. Oracle Solaris 11.4 builds on this validated foundation and includes software improvements that address performance, functionality, and reliability. Whenever possible, you should configure Oracle Solaris 11.4 in FIPS 140-2 mode to take advantage of these improvements.

The following mechanisms are available to IPsec and approved for use in Oracle Solaris in FIPS 140-2 mode:

- AES in CBC, CCM, and GCM modes in 128-bit to 256-bit key lengths
- 3DES
- SHA1
- SHA2 in 256-bit to 512-bit key lengths

For the definitive list of FIPS 140-2 approved algorithms for Oracle Solaris, follow the links in "FIPS 140-2 Level 1 Guidance Documents for Oracle Solaris Systems" in *Using a FIPS 140-2 Enabled System in Oracle Solaris 11.4*.

## IPsec and NAT Traversal

IKE can negotiate IPsec SAs across a NAT box. This ability enables systems to securely connect from a remote network even when the systems are behind a NAT device. For example, employees who work from home or who log on from a conference site can protect their traffic with IPsec.

A NAT box translates a private internal address into a unique Internet address. NATs are very common at public access points to the Internet, such as hotels.

The ability to use IKE when a NAT box is between communicating systems is called "NAT traversal", or NAT-T. NAT-T has the following limitations:

- The AH protocol depends on an unchanging IP header, therefore, AH cannot work with NAT-T. The ESP protocol is used with NAT-T.
- The NAT box does not use special processing rules. A NAT box with special IPsec processing rules might interfere with the implementation of NAT-T.
- NAT-T works only when the IKE initiator is the system behind the NAT box. An IKE responder cannot be behind a NAT box unless the box has been programmed to forward IKE packets to the appropriate individual system behind the box.

The following RFCs describe NAT functionality and the limits of NAT-T. Copies of the RFCs are available at `https://www.rfc-editor.org`.

- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," January 2001
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," March 2004
- RFC 3947, "Negotiation of NAT-Traversal in the IKE," January 2005
- RFC 3948, "UDP Encapsulation of IPsec Packets," January 2005

## IPsec and SCTP

Oracle Solaris supports the Streams Control Transmission Protocol (SCTP). The use of the SCTP protocol and SCTP port number to specify IPsec policy is supported, but is not robust. The IPsec extensions for SCTP as specified in RFC 3554 are not yet implemented. These limitations can create complications in creating IPsec policy for SCTP.

SCTP can make use of multiple source and destination addresses in the context of a single SCTP association. When IPsec policy is applied to a single source or a single destination address, communication can fail when SCTP switches the source or the destination address of that association. IPsec policy only recognizes the original address. For information about SCTP, read the RFC Stream Control Transmission Protocol (SCTP).

## IPsec and Oracle Solaris Zones

IPsec is supported in Oracle Solaris Zones called exclusive-IP zones. Every zone can have its own IPsec policy and IKE configuration and is treated like a separate host.

Shared-IP zones in Trusted Extensions do not support IPsec per zone because these zones do not have their own IP stack. For shared-IP zones, the IPsec policy and IKE configuration are performed in the global zone. The IPsec policy rules for the shared-IP zone are the rules for the zone's shared IP address.

For more information, see Chapter 1, “Oracle Solaris Zones Introduction” in *Introduction to Oracle Solaris Zones* and “Access to Labeled Zones” in *Trusted Extensions Configuration and Administration*.

## IPsec and Virtual Machines

IPsec works with virtual machines (VMs). To create VMs on SPARC systems, use the Oracle VM Server for SPARC. On x86 systems, you can use the Oracle VM Server for x86. For information about configuration, see the administration guide for the version of your Oracle VM.

## IPsec Configuration Commands and Files

Table 6, "Selected IPsec Configuration Commands and Files," on page 101 describes the files, commands, and service identifiers that are used to configure and manage IPsec. For completeness, the table includes key management files, socket interfaces, and commands.

For more information about service identifiers, see Chapter 1, "Introduction to the Service Management Facility" in *Managing System Services in Oracle Solaris 11.4*.

For instructions on implementing IPsec on your network, see "Protecting Network Traffic With IPsec" on page 103.

For more details about IPsec utilities and files, see Chapter 12, "IPsec and Key Management Reference".

**TABLE 6** Selected IPsec Configuration Commands and Files

| IPsec Command, File, or Service | Description | Man Page |
|---|---|---|
| `svc:/network/ipsec/ipsecalgs` | The SMF service that manages IPsec algorithms. | `ipsecalgs`(8) |
| `svc:/network/ipsec/manual-key` | The SMF service that manages manually keyed IPsec SAs. | `ipseckey`(8) |
| `svc:/network/ipsec/policy` | The SMF service that manages IPsec policy. | `smf`(7), `ipsecconf`(8) |
| `svc:/network/ipsec/ike:ikev2`, `svc:/network/ipsec/ike:default` | The SMF service instances for the automatic management of IPsec SAs by using IKE. | `smf`(7), `in.ikev2d`(8), `in.iked`(8) |
| `/etc/inet/ipsecinit.conf` file | IPsec policy file.<br><br>Used by the SMF `policy` service to configure IPsec policy at system boot. | `ipsecconf`(8) |
| `ipsecconf` command | IPsec policy command. Useful for viewing and modifying the current IPsec policy, and for testing.<br><br>Used by the SMF `policy` service to configure IPsec policy at system boot. | `ipsecconf`(8) |

| IPsec Command, File, or Service | Description | Man Page |
|---|---|---|
| PF_KEY socket interface | Interface for the security associations database (SADB). Handles manual key management and automatic key management. | pf_key(4P) |
| ipseckey command | IPsec SAs keying command. ipseckey is a command-line front end to the PF_KEY interface. ipseckey can create, destroy, or modify SAs. | ipseckey(8) |
| /etc/inet/secret/ipseckeys file | Contains manually keyed SAs.<br><br>Used by the SMF manual-key service to configure SAs manually at system boot. | |
| ipsecalgs command | IPsec algorithms command. Useful for viewing and modifying the list of IPsec algorithms and their properties.<br><br>Used by the SMF ipsecalgs service to synchronize known IPsec algorithms with the kernel at system boot. | ipsecalgs(8) |
| /etc/inet/ipsecalgs file | Contains the configured IPsec mechanisms and algorithm definitions. This file is managed by the ipsecalgs command and must never be edited manually. | |
| /etc/inet/ike/ikev2.config file | IKEv2 configuration and policy file. Key management is based on rules and global parameters from this file. See "IKEv2 Utilities and Files" on page 223. | ikev2.config(5) |
| /etc/inet/ike/config file | IKEv1 configuration and policy file. By default, this file does not exist. Key management is based on rules and global parameters from this file. See "IKEv1 Utilities and Files" on page 227.<br><br>If this file exists, the svc:/network/ipsec/ike:default service starts the IKEv1 daemon, in.iked. | ike.config(5) |

♦ ♦ ♦ C H A P T E R 7

# Configuring IPsec

This chapter provides procedures for implementing IPsec on your network. The procedures are described in the following sections:

- "Protecting Network Traffic With IPsec" on page 103
- "Protecting a VPN With IPsec" on page 112
- "Additional IPsec Tasks" on page 120

For overview information about IPsec, see Chapter 6, "About IP Security Architecture". For reference information about IPsec, see Chapter 12, "IPsec and Key Management Reference".

## Protecting Network Traffic With IPsec

The procedures in this section enable you to secure traffic between two systems and to secure a web server. To protect a VPN, see "Protecting a VPN With IPsec" on page 112. For additional procedures to manage IPsec and to use SMF commands with IPsec and IKE, see "Additional IPsec Tasks" on page 120.

The following information applies to all IPsec configuration tasks:

- **IPsec and zones –** Each system is either a global zone or an exclusive-IP zone. For more information, see "IPsec and Oracle Solaris Zones" on page 100.
- **IPsec and FIPS 140-2 mode –** As the IPsec administrator, you are responsible for choosing algorithms that are FIPS 140-2 approved for Oracle Solaris. The procedures and examples in this chapter use FIPS 140-2 approved algorithms except when the algorithm "`any`" is specified.
- **IPsec and RBAC –** To use roles to administer IPsec, see Chapter 3, "Assigning Rights in Oracle Solaris" in *Securing Users and Processes in Oracle Solaris 11.4*. For an example, see "How to Configure a Role for Network Security" on page 123.
- **IPsec and SCTP –** You can use IPsec to protect Streams Control Transmission Protocol (SCTP) associations, but caution must be used. For more information, see "IPsec and SCTP" on page 100.

- **IPsec and Trusted Extensions labels –** On systems that are configured with the Trusted Extensions feature of Oracle Solaris, labels can be added to IPsec packets. For more information, see "Administration of Labeled IPsec" in *Trusted Extensions Configuration and Administration*.
- **IPv4 and IPv6 addresses –** The IPsec examples in this guide use IPv4 addresses. Oracle Solaris supports IPv6 addresses as well. To configure IPsec for an IPv6 network, substitute IPv6 addresses in the examples. When protecting tunnels with IPsec, you can mix IPv4 and IPv6 addresses for the inner and outer addresses. This type of a configuration enables you to tunnel IPv6 over an IPv4 network, for example.

The following task map lists procedures that set up IPsec between one or more systems. The `ipsecconf`(8), `ipseckey`(8), and `ipadm`(8) man pages also describe useful procedures in their respective Examples sections.

**TABLE 7** Protecting Network Traffic With IPsec Task Map

| Task | Description | For Instructions |
|---|---|---|
| Secure traffic between two systems. | Protects packets from one system to another system. | "How to Secure Network Traffic Between Two Servers With IPsec" on page 105 |
| Configure IPsec remotely. | Uses the `ssh` command to reach remote systems and configure them with IPsec. | Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 |
| Configure IPsec for a system that is running in FIPS 140-2 mode. | Selects only FIPS 140-2 algorithms for IPsec. | Example 20, "Configuring IPsec Policy With FIPS 140-2 Approved Algorithms," on page 108 |
| Specify the IKE protocol version to use for an IPsec rule. | Helps in transitioning to an all-IKEv2 network. | Example 21, "Configuring IPsec Policy to Use the IKEv2 Protocol Only," on page 108 |
| Use the `or pass` action in an IPsec rule. | Helps when transitioning to a network where all systems are protected by IPsec. | Example 22, "Transitioning Client Systems to Use IPsec by Using the `or pass` Action on the Server," on page 109 |
| Secure a web server by using IPsec policy. | Requires non-web traffic to use IPsec. Web clients are identified by particular ports that bypass IPsec checks. | "How to Use IPsec to Protect Web Server Communication With Other Servers" on page 109 |
| Use IKE to automatically create keying material for IPsec SAs. | Recommended method of creating IPsec SAs. | "Configuring IKEv2" on page 141 and "Configuring IKEv1" on page 167 |
| Set up a secure virtual private network (VPN). | Sets up IPsec between two systems across the Internet. | "Protecting a VPN With IPsec" on page 112 |
| Set up manual key management. | Provides the raw data for IPsec SAs without using IKE. | "How to Manually Create IPsec Keys" on page 121 |

## ▼ How to Secure Network Traffic Between Two Servers With IPsec

This procedure assumes the following setup:

- The systems are assigned static IP addresses. For more information, see the `netcfg(8)` man page.
- The two systems are named `host1` and `host2`.
- Each system has an IP address. This can be an IPv4 address, an IPv6 address, or both. This procedure uses IPv4 addresses.
- Each system is either a global zone or an exclusive-IP zone. For more information, see "IPsec and Oracle Solaris Zones" on page 100.
- Each system encrypts traffic with the AES algorithm and authenticates it with SHA-2.

  **Note -** The SHA-2 algorithm might be required at some sites.

- Each system uses shared security associations.

  With shared SAs, only one pair of SAs is needed to protect the two systems.

**Note -** To use IPsec with labels on a Trusted Extensions system, see the extension of this procedure in "How to Apply IPsec Protections in a Multilevel Trusted Extensions Network" in *Trusted Extensions Configuration and Administration*.

**Before You Begin** A user with specific rights can run the following commands without being `root`:

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- In this administrative role, you can edit IPsec-related system files and create keys by using the `pfedit` command.
- To edit the `hosts` file, you must be in the `root` role or have explicit permission to edit that file. See Example 27, "Enabling a Trusted User to Configure and Manage IPsec," on page 125.

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **On each system, add host entries to the `/etc/inet/hosts` file.**

   This step enables the local naming service to resolve system names to IP addresses without depending on a networked naming service.

   a. **On a system that is named `host2`, type the following in the `hosts` file:**

   ```
   ## Secure communication with host1
   198.51.100.6 host1
   ```

   b. **On a system that is named `host1`, type the following in the `hosts` file:**

   ```
   ## Secure communication with sytem2
   198.51.100.33 host2
   ```

2. **On each system, create the IPsec policy file.**

   The file name is `/etc/inet/ipsecinit.conf`. For an example, see the `/etc/inet/ipsecinit.sample` file.

   ```
   # pfedit /etc/inet/ipsecinit.conf
   ```

3. **Add an IPsec policy entry to the `ipsecinit.conf` file.**

   For the syntax of IPsec policy entries and several examples, see the `ipsecconf(8)` man page.

   a. **On the `host1` system, add the following policy:**

   ```
   {laddr host1 raddr host2} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

   Because the `dir` keyword is not used, the policy applies to both outbound and inbound packets.

   b. **On the `host2` system, add the identical policy:**

   ```
   {laddr host2 raddr host1} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

4. **On each system, configure IKE to manage the IPsec SAs.**

   Follow one of the configuration procedures in "Configuring IKEv2" on page 141. For the syntax of the IKE configuration file, see the `ikev2.config(5)` man page. If you are communicating with a system that only supports the IKEv1 protocol, refer to "Configuring IKEv1" on page 167 and the `ike.config(5)` man page.

   **Note -** If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 121.

5. **Verify the syntax of the IPsec policy file.**

   ```
   $ pfbash
   $ /usr/sbin/ipsecconf -c /etc/inet/ipsecinit.conf
   ```

   Fix any errors, verify the syntax of the file, and continue.

6. **Refresh the IPsec policy.**

   ```
   $ svcadm refresh ipsec/policy:default
   ```

   IPsec policy is enabled by default, so you *refresh* it. If you have disabled the IPsec policy, enable it.

   ```
   $ svcadm enable ipsec/policy:default
   ```

7. **Activate the keys for IPsec.**

   - **If the `ike` service is not enabled, enable it.**

   **Note -** If you are communicating with a system that can only run the IKEv1 protocol, specify the `ike:default` instance.

   ```
   $ svcadm enable ipsec/ike:ikev2
   ```

   - **If the `ike` service is enabled, restart it.**

   ```
   $ svcadm restart ike:ikev2
   ```

   If you manually configured keys in Step 4, complete the procedure "How to Manually Create IPsec Keys" on page 121 to activate the keys.

8. **Verify that packets are being protected.**

   For the procedure, see "How to Verify That Packets Are Protected With IPsec" on page 127.

**Example 19** Configuring IPsec Policy Remotely by Using an `ssh` Connection

In this example, the administrator in the `root` role configures IPsec policy and keys on two systems by using the `ssh` command to reach the second system. The administrator is defined identically on both systems. For more information, see the `ssh`(1) man page.

1. The administrator configures the first system by performing Step 1 through Step 5 of "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.
2. In a different terminal window, the administrator uses the identically defined user name and ID to log in remotely with the `ssh` command.

```
local-system $ ssh -l jdoe other-system
other-system $ su - root
Enter password: xxxxxxxx
other-system #
```

3. In the terminal window of the `ssh` session, the administrator configures the IPsec policy and keys of the second system by completing Step 1 through Step 7.
4. The administrator ends the `ssh` session.

```
other-system # exit
local-system
$ exit
```

5. The administrator enables IPsec policy on the first system by completing Step 6 and Step 7.

The next time the two systems communicate, including by using an `ssh` connection, the communication is protected by IPsec.

**Example 20** Configuring IPsec Policy With FIPS 140-2 Approved Algorithms

In this example, the administrator configures the IPsec policy on a FIPS 140-2-enabled system to follow a site security policy that requires symmetric algorithms whose key length is 256 bits.

The administrator specifies two possible IPsec policies. The first policy specifies AES in CCM mode with a 256-bit key length for encryption. The second policy specifies AES with a 256-bit key length for encryption and SHA384 for authentication.

```
 {laddr host1 raddr host2} ipsec {encr_algs aes-ccm(256) sa shared} or ipsec
 {laddr host1 raddr host2} ipsec {encr_algs aes(256) encr_auth_algs sha384 sa shared}
```

**Example 21** Configuring IPsec Policy to Use the IKEv2 Protocol Only

In this example, the administrator configures the IPsec policy on a server to ignore the IKEv1 protocol. All SAs will be created by IKEv2. Attempted negotiations with IKEv1 will fail. The administrator creates a corresponding IKEv2 configuration file.

```
## ipsecinit.conf
{raddr 192.0.2.0/27 dir both } ipsec {encr_algs aes-ccm sa shared ike_version 2}
```

This rule says that any host on the `192.0.2.0/27` subnet can talk to the server by using the combined mode `aes-ccm` algorithm.

The corresponding IKEv2 configuration file enables all address ranges whose certificate is signed by the same chain of trust as the server to connect with the `server1.example.com` server:

```
## ikev2.config
        ikesa_xform { dh_group 21 auth_alg sha256 encr_alg aes }

{
        label "IKEv2-only certificate"

        request_http_certs yes
        auth_method cert
        local_id DNS = "server1.example.com"
        remote_id ANY
        local_addr 0.0.0.0/0
        remote_addr 0.0.0.0/0
}
```

The configuration of the `server1` server is complete. Client systems who install the certificate and configure IPsec and IKEv2 can communicate with `server1`.

**Example 22** Transitioning Client Systems to Use IPsec by Using the `or pass` Action on the Server

In this example, the administrator is gradually configuring all clients on a subnet to use IPsec. The `or pass {}` action enables the server to receive packets from all clients on the subnet, including clients that are not configured with IPsec.

```
## ipsecinit.conf
{raddr 192.0.2.0/27 dir both } ipsec {encr_algs aes-ccm sa shared} or pass {}
```

The `or pass {}` action passes all traffic that is not encrypted by the specified encryption mechanisms but otherwise satisfies the rule, as well as IPsec traffic that matches the rule. The connection to the server that has an `or pass` rule must originate from the client. The action is cached when the connection is first established.

All clients on the `192.0.2.0` subnet will be able to establish a connection to the server.

## ▼ How to Use IPsec to Protect Web Server Communication With Other Servers

On a system that runs a web server, you can use IPsec to protect all traffic except web client requests. The protected network traffic is typically between the web server and other backend servers.

In addition to allowing web clients to bypass IPsec, the IPsec policy in this procedure allows the server to make DNS client requests. All other traffic is protected by IPsec.

**Before You Begin** This procedure assumes that the steps in "How to Secure Network Traffic Between Two Servers With IPsec" on page 105 that configure IPsec on your two servers have been completed so that the following conditions are in effect:

- Each system is either a global zone or an exclusive-IP zone with a fixed address. For more information, see "IPsec and Oracle Solaris Zones" on page 100.
- Communication with the web server is already protected by IPsec.
- Keying material is being generated by IKE.
- You have verified that packets are being protected.

A user with specific rights can run these commands without being `root`.

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- To edit IPsec-related system files and create keys, you use the `pfedit` command.
- To edit the `hosts` file, you must be in the `root` role or have explicit permission to edit that file.

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Determine which services need to bypass IPsec policy checks.**

   For a web server, these services include TCP ports 80 (HTTP) and 443 (Secure HTTP). If the web server provides DNS name lookups, the server might also need to include port 53 for both TCP and UDP.

2. **Add the web server policy to the IPsec policy file.**

   Add the following lines to the `ipsecinit.conf` file:

```
# pfedit /etc/inet/ipsecinit.conf
...
# Web traffic that web server should bypass.
{lport  80 ulp tcp dir both} bypass {}
{lport 443 ulp tcp dir both} bypass {}

# Outbound DNS lookups should also be bypassed.
{rport 53 dir both} bypass {}

# Require all other traffic to use ESP with AES and SHA-2.
# Use a unique SA for outbound traffic from the port
```

```
{} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

This configuration allows only secure traffic to access the system, with the bypass exceptions that are described in Step 1.

3. **Verify the syntax of the IPsec policy file.**

   ```
   # ipsecconf -c /etc/inet/ipsecinit.conf
   ```

4. **Refresh the IPsec policy.**

   ```
   # svcadm refresh ipsec/policy
   ```

5. **Refresh the keys for IPsec.**

   Restart the `ike` service.

   ```
   # svcadm restart ike:ikev2
   ```

   **Note -** If you are communicating with a system that can only run the IKEv1 protocol, specify the `ike:default` instance.

   If you manually configured the keys, follow the instructions in "How to Manually Create IPsec Keys" on page 121.

   Your setup is complete.

6. **(Optional) Enable a remote system to communicate with the web server for nonweb traffic.**

   Add the following lines to a remote system's `/etc/inet/ipsecinit.conf` file:

   ```
   ## Communicate with web server about nonweb stuff
   ##
   {raddr webserver} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

   Verify the syntax and then refresh the IPsec policy to activate it.

   ```
   remote-system # ipsecconf -c /etc/inet/ipsecinit.conf
   remote-system # svcadm refresh ipsec/policy
   ```

   A remote system can communicate securely with the web server for nonweb traffic only when the systems' IPsec policies match.

7. **(Optional) Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.**

   ```
   # ipsecconf -L -n
   ```

## Protecting a VPN With IPsec

You can use IPsec to protect a VPN. For background, see "Transport and Tunnel Modes in IPsec" on page 95. The examples and procedures in this section use IPv4 addresses, but the examples and procedures apply to IPv6 VPNs as well. For a short discussion, see "Protecting Network Traffic With IPsec" on page 103.

For examples of IPsec policies for tunnel mode, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 112.

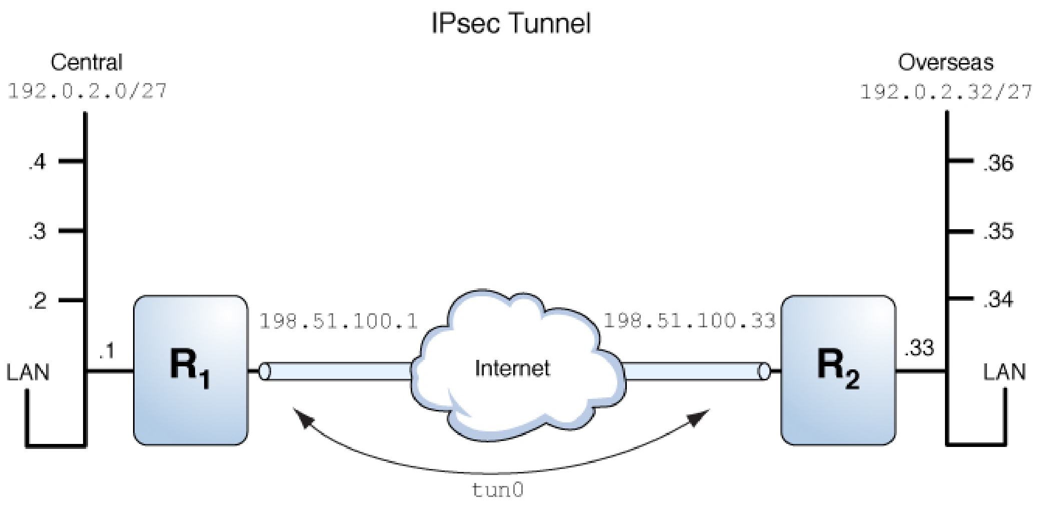
### Examples of Protecting a VPN With IPsec by Using Tunnel Mode

The tunnel in the following illustration is configured for all subnets of the LANs as follows:

```
## Tunnel configuration for ##
# Tunnel name is tun0
# Intranet point for the source is 192.0.2.1/
# Intranet point for the destination is 192.0.2.44
# Tunnel source is 198.51.100.1
# Tunnel destination is 198.51.100.33

# Tunnel name address object is tun0/to-central
# Tunnel name address object is tun0/to-overseas
```

**FIGURE 9** Tunnel Protected by IPsec



The following examples are based on the illustration.

**EXAMPLE 23** Creating a Tunnel That All Subnets Can Use

In this example, all traffic from the local LANs of the Central LAN in Figure 9, "Tunnel Protected by IPsec," on page 113 can be tunneled through Router 1 to Router 2, and then delivered to all local LANs of the Overseas LAN. The traffic is encrypted with AES.

```
## IPsec policy ##
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

**EXAMPLE 24** Creating a Tunnel That Connects Two Subnets Only

In this example, only traffic between subnet `192.0.2.0/27` of the Central LAN and subnet `192.0.2.32/27` of the Overseas LAN is tunneled and encrypted. In the absence of other IPsec policies for Central, if the Central LAN attempts to route any traffic for other LANs over this tunnel, the traffic is dropped at Router 1.

```
## IPsec policy ##
{tunnel tun0 negotiate tunnel laddr 192.0.2.0/27 raddr 192.0.2.32/27}
```

```
ipsec {encr_algs aes encr_auth_algs sha512 shared}
```

## Description of the Network Topology for the IPsec Tasks to Protect a VPN

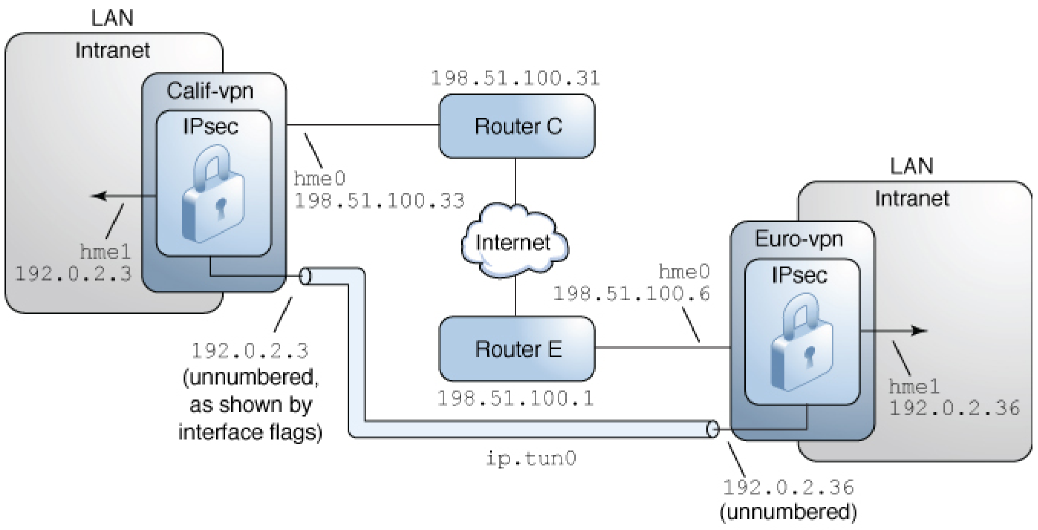
The procedures in this section assume the following setup. For a depiction of the network, see Figure 10, “Sample VPN Between Offices Connected Across the Internet,” on page 115.

- Each system is using an IPv4 address space.

  These procedures also work with IPv6 addresses or a combination of IPv4 and IPv6 addresses.
- Each system has two interfaces. The `net0` interface connects to the Internet. In this example, Internet IP addresses begin with `198.51.100`. The `net1` interface connects to the company's LAN, its intranet. In this example, intranet IP addresses begin with the number `192.0.2`.
- Each system requires ESP encryption with the AES algorithm. The AES algorithm uses a 128-bit or 256-bit key.
- Each system requires ESP authentication with the SHA-2 algorithm. In this example, the SHA-2 algorithm uses a 512-bit key.
- Each system can connect to a router that has direct access to the Internet.
- Each system uses shared security associations.

The following illustration shows the configuration parameters used in the procedures.

**FIGURE 10** Sample VPN Between Offices Connected Across the Internet



The configuration parameters are listed in the following table.

| Parameter | Europe | California |
|---|---|---|
| System name | `euro-vpn` | `calif-vpn` |
| System intranet interface | `net1` | `net1` |
| System intranet address, the default route to the other network | `192.0.2.36` | `192.0.2.3` |
| System intranet address object | `net1/inside` | `net1/inside` |
| System Internet interface | `net0` | `net0` |
| System Internet address | `198.51.100.6` | `198.51.100.33` |
| Name of Internet router | `router-E` | `router-C` |
| Address of Internet router | `198.51.100.1` | `198.51.100.31` |
| Tunnel name | `tun0` | `tun0` |
| Tunnel name address object | `tun0/v4tunaddr` | `tun0/v4tunaddr` |

For information about tunnel names, see "Administering IP Tunnels" in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*. For information about address objects, see "How to Configure an IPv4 Interface" in *Configuring and Managing Network Components in Oracle Solaris 11.4* and the `ipadm(8)` man page.

## ▼ How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode

In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

This procedure extends the procedure "How to Secure Network Traffic Between Two Servers With IPsec" on page 105. The setup is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 114.

For a fuller description of the reasons for running particular commands, see the corresponding steps in "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

---

**Note -** Perform the steps in this procedure on both systems.

---

In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

---

**Note -** To use IPsec in tunnel mode with labels on a Trusted Extensions system, see the extension of this procedure in "How to Configure a Tunnel Across an Untrusted Network" in *Trusted Extensions Configuration and Administration*.

---

**Before You Begin** Each system is either a global zone or an exclusive-IP zone. For more information, see "IPsec and Oracle Solaris Zones" on page 100.

A user with specific rights can run these commands without being `root`.

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- To edit IPsec-related system files and create keys, you use the `pfedit` command.
- To edit the `hosts` file, you must be in the `root` role or have explicit permission to edit that file.

For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Control the flow of packets before configuring IPsec.**

   a. **Disable IP forwarding and IP dynamic routing.**

Follow the instructions in "How to Enable Dynamic Routing on a Single-Interface System" in *Configuring an Oracle Solaris 11.4 System as a Router or a Load Balancer*.

Disabling IP forwarding prevents packets from being forwarded from one network to another network through this system. For a description of the `ipadm` command, see the `ipadm`(8) man page.

b. **Enable IP strict multihoming.**

```
# ipadm set-prop -p hostmodel=strong ipv4
```

Enabling IP strict multihoming requires that packets for one of the system's destination addresses arrive at the correct destination address.

When the `hostmodel` parameter is set to `strong`, packets that arrive on a particular interface must be addressed to one of the local IP addresses of that interface. All other packets, even packets that are addressed to other local addresses of the system, are dropped.

c. **Verify that most network services are disabled.**

Verify that the `ssh` service is running.

```
$ svcs | grep network
…
online         Aug_09   svc:/network/ssh:default
```

2. **Add the IPsec policy for the VPN to the `/etc/inet/ipsecinit.conf` file.**

For additional examples, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 112.

In this policy, IPsec protection is not required between systems on the local LAN and the internal IP address of the gateway, so a `bypass` statement is added.

a. **On the `euro-vpn` system, add the following entry to the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 192.0.2.36 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-2.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

b. **On the `calif-vpn` system, add the following entry to the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
```

```
{laddr 192.0.2.3 dir both} bypass {}

# WAN traffic uses ESP with AES and SHA-2.
{tunnel tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

3. **On each system, configure IKE to add a pair of IPsec SAs between the two systems.**

   Configure IKE by following one of the configuration procedures in "Configuring IKEv2" on page 141. For the syntax of the IKE configuration file, see the `ikev2.config`(5) man page. If you are communicating with a system that only supports the IKEv1 protocol, refer to "Configuring IKEv1" on page 167 and the `ike.config`(5) man page.

   **Note -** If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 121.

4. **Verify the syntax of the IPsec policy file.**

   ```
   # ipsecconf -c /etc/inet/ipsecinit.conf
   ```

   Fix any errors, verify the syntax of the file, and continue.

5. **Refresh the IPsec policy.**

   ```
   # svcadm refresh ipsec/policy
   ```

   IPsec policy is enabled by default, so you *refresh* it. If you have the disabled IPsec policy, enable it.

   ```
   # svcadm enable ipsec/policy
   ```

6. **Create and configure the tunnel, `tun0`.**

   The following commands configure the internal and external interfaces, create the `tun0` tunnel, and assign IP addresses to the tunnel.

   a. **On the `calif-vpn` system, create the tunnel and configure it.**

      ```
      # ipadm create-ip net1
      # ipadm create-addr -T static -a local=192.0.2.3 net1/inside
      # dladm create-iptun -T ipv4 -a local=198.51.100.33,remote=198.51.100.6 tun0
      # ipadm create-ip tun0
      # ipadm create-addr -T static \
      -a local=192.0.2.3,remote=192.0.2.36 tun0/v4tunaddr
      ```

The first command creates the IP interface `net1`. The second command adds addresses to `net1`. The third command creates the IP interface `tun0`. The fourth command adds IP addresses that are encapsulated in the tunnel link. For more information, see the `dladm`(8) and `ipadm`(8) man pages.

b. **On the `euro-vpn` system, create the tunnel and configure it.**

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=192.0.2.36 net1/inside
# dladm create-iptun -T ipv4 -a local=198.51.100.6,remote=198.51.100.33 tun0
# ipadm create-ip tun0
# ipadm create-addr -T static \
-a local=192.0.2.36,remote=192.0.2.3 tun0/v4tunaddr
```

---

**Note -** The `-T` option to the `ipadm` command specifies the type of address to create. The `-T` option to the `dladm` command specifies the tunnel.

---

For information about these commands, see the `dladm`(8) and `ipadm`(8) man pages, and "How to Configure an IPv4 Interface" in *Configuring and Managing Network Components in Oracle Solaris 11.4*.

7. **On each system, configure forwarding.**

```
# ipadm set-ifprop -m ipv4 -p forwarding=on net1
# ipadm set-ifprop -m ipv4 -p forwarding=on tun0
# ipadm set-ifprop -m ipv4 -p forwarding=off net0
```

IP forwarding means that packets that arrive from somewhere else can be forwarded. IP forwarding also means that packets that leave this interface might have originated somewhere else. To successfully forward a packet, both the receiving interface and the transmitting interface must have IP forwarding enabled.

Because the `net1` interface is *inside* the intranet, IP forwarding must be enabled for `net1`. Because `tun0` connects the two systems through the Internet, IP forwarding must be enabled for `tun0`. The `net0` interface has its IP forwarding disabled off to prevent an *outside* adversary on the Internet from injecting packets into the protected intranet.

8. **On each system, prevent the advertising of the private interface.**

```
# ipadm set-addrprop -p private=on net0
```

Even if `net0` has IP forwarding disabled, a routing protocol implementation might still advertise the interface. For example, the `in.routed` protocol might still advertise that `net0` is available to forward packets to its peers inside the intranet. By setting the interface's `private` flag, these advertisements are prevented.

9. **Restart the network services.**

   ```
   # svcadm restart svc:/network/initial:default
   ```

10. **Manually add a default route over the `net0` interface.**

    The default route must be a router with direct access to the Internet.

    a. **On the `calif-vpn` system, add the following route:**

    ```
    # route -p add net default 198.51.100.31
    ```

    b. **On the `euro-vpn` system, add the following route:**

    ```
    # route -p add net default 198.51.100.1
    ```

    Even though the `net0` interface is not part of the intranet, `net0` does need to reach across the Internet to its peer system. To find its peer, `net0` needs information about Internet routing. The VPN system appears to be a host, rather than a router, to the rest of the Internet. Therefore, you can use a default router or run the router discovery protocol to find a peer system. For more information, see the `route`(8) and `in.routed`(8) man pages.

## Additional IPsec Tasks

The following task map lists tasks that you might use when managing IPsec.

**TABLE 8** Additional IPsec Tasks Task Map

| Task | Description | For Instructions |
|---|---|---|
| Create or replace IPsec SAs manually. | Provides the raw data for IPsec SAs: | "How to Manually Create IPsec Keys" on page 121 |
| Create a Network Security role. | Creates a role that can set up a secure network, but has fewer powers than the `root` role. | "How to Configure a Role for Network Security" on page 123 |
| Create a rights profile that can handle all network management tasks. | Creates a role that can perform network management but has fewer powers than the `root` role. | Example 27, "Enabling a Trusted User to Configure and Manage IPsec," on page 125 |
| Check that IPsec is protecting the packets. | Examines `snoop` output for specific headers that indicate how the IP packets are protected. | "How to Verify That Packets Are Protected With IPsec" on page 127 |
| Manage IPsec and keying material as a set of SMF services. | Enables, disables, refreshes, and restarts services. Also changes the property values of services. | "Viewing IPsec and Manual Key Service Properties" on page 207 |

## ▼ How to Manually Create IPsec Keys

The following procedure provides the IPsec keys for when you are not using only IKE for key management.

IPsec SAs that are added by using the `ipseckey` command are not persistent over system reboot. For persistent IPsec SAs, add entries to the `/etc/inet/secret/ipseckeys` file.

**Caution -** If you must use manual keying, take great care to ensure that the keys that you generate are secure. These are the actual keys used to secure the data.

**Before You Begin** You must be in the global zone to manually manage keying material in a shared-IP zone. For an exclusive-IP zone, you configure the keying material in that exclusive-IP zone.

You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Generate the keys for the IPsec SAs.**

   The keys must support a specific policy in the `ipsecinit.conf` file. For example, you might use the policy from "How to Secure Network Traffic Between Two Servers With IPsec" on page 105:

   ```
   {laddr host1 raddr host2} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

   This policy uses the AES and SHA-2 algorithms.

   a. **Determine the keys that you require.**

      You need to generate keys for `aes`, `sha512`, and the security parameter index (SPI) for the SA:

      - Two hexadecimal random numbers as the value for the SPI. One number is for outbound traffic. One number is for inbound traffic. Each number can be up to eight characters long.
      - Two hexadecimal random numbers for the SHA-2 authentication algorithm. Each number must be 512 characters long. One number is for `dst host1`. One number is for `dst host2`.
      - Two hexadecimal random numbers for the AES encryption algorithm. Each number must be 128 characters long. One number is for `dst host1`. One number is for `dst host2`.

---

**Note -** The `ipsecalgs -l` command displays the key sizes of the algorithms. Follow this procedure when using manual keys, that is, use the SHA512 and AES algorithms. Do not use weak algorithms, the combined mode algorithms, or the GMAC algorithms for manual keys.

---

b. **Generate the required keys.**

- If you have a random number generator at your site, use the generator.
- Use the `pktool` command, as shown in "How to Generate a Symmetric Key by Using the pktool Command" in *Managing Encryption and Certificates in Oracle Solaris 11.4* and the IPsec example in that section.

2. **Add the keys to the manual keys file for IPsec.**

a. **Edit the `/etc/inet/secret/ipseckeys` file on the `host1` system to appear similar to the following:**

```
## ipseckeys - This file takes the file format documented in
##  ipseckey(8).
#   Note that naming services might not be available when this file
#   loads, just like ipsecinit.conf.
#
#   Backslashes indicate command continuation.
#
# for outbound packets on host1
add esp spi 0x8bcd1407 \
   src 198.51.100.6 dst 198.51.100.33  \
   encr_alg aes \
   auth_alg sha512  \
   encrkey  abcdefabcdefabcdefabcdefabcdefab... \
   authkey  123456781284567891234567891234\u0035\u0036...
#
# for inbound packets
add esp spi 0xnnnnnnnn \
   src 198.51.100.33 dst 198.51.100.6 \
   encr_alg aes \
   auth_alg sha512  \
   encrkey fedcbafedcbafedcbafedcbafedcbafe... \
   authkey 987654321234567898765432123456\u0037\u0038...
```

b. **Protect the file with read-only permissions.**

```
# chmod 400 /etc/inet/secret/ipseckeys
```

If you used the `pfedit -s` command to create the `ipseckeys` file, then the permissions are correctly set. For more information, see the `pfedit(8)` man page.

c. **Verify the syntax of the file.**

```
# ipseckey -c /etc/inet/secret/ipseckeys
```

---

**Note -** The keys on the two systems *must* be identical.

---

3. **Activate the keys for IPsec.**

   - **If the `manual-key` service is not enabled, enable it.**

   ```
   $ svcs manual-key
   STATE          STIME    FMRI
   disabled       Apr_10   svc:/network/ipsec/manual-key:default
   # svcadm enable ipsec/manual-key
   ```

   - **If the `manual-key` service is enabled, refresh it.**

   ```
   # svcadm refresh ipsec/manual-key
   ```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Configure a Role for Network Security

If you are using the rights feature of Oracle Solaris to administer your systems, use this procedure to provide a network management role or network security role.

**Before You Begin** You must assume the `root` role to create and assign a role. Regular users can list and view the contents of available rights profiles.

1. **List the available network-related rights profiles.**

```
$ getent prof_attr | grep Network | more
...
Network Management:RO::Manage the host and network configuration...
Network Security:RO::Manage network and host security...:profiles=Network Wifi
```

```
Security,Network Link Security,Network IPsec Management...
Network Wifi Management:RO::Manage wifi network configuration...
Network Wifi Security:RO::Manage wifi network security...
Network Link Security:RO::Manage network link security...
Network IPsec Management:RO::Manage IPsec and IKE...
System Administrator:RO::Can perform most non-security administrative tasks:
profiles=...Network Management...
Information Security:RO::Maintains MAC and DAC security policies:
profiles=...Network Security...
```

The Network Management profile is a supplementary profile in the System Administrator profile. If you have included the System Administrator rights profile in a role, then that role can execute the commands in the Network Management profile.

2. **List the commands in the Network Management rights profile.**

```
$ profiles -p "Network Management" info
...
cmd=/usr/sbin/dladm
cmd=/usr/sbin/dlstat
...
cmd=/usr/sbin/svcadm
cmd=/usr/sbin/svccfg
cmd=/usr/sbin/dumpcap
```

3. **Decide the scope of the network security roles at your site.**

   Use the definitions of the rights profiles in Step 1 to guide your decision.

   - To create a role that handles all network security, use the Network Security rights profile.
   - To create a role that handles IPsec and IKE only, use the Network IPsec Management rights profile.
   - To create a role that handles network management and security, use the Network Security or the Network IPsec Management rights profile, in addition to the Network Management profile.

4. **Create the role and assign the role to one or more users.**

   For the steps, see "Creating a Role" in *Securing Users and Processes in Oracle Solaris 11.4* and Example 27, "Enabling a Trusted User to Configure and Manage IPsec," on page 125.

**Example 25** Creating and Assigning a Network Management and Security Role

In this example, the administrator assigns to a role two rights profiles, Network Management and Network Security. Then the administrator assigns the role to a trusted user.

```
# roleadd -c "Network Mgt and Security" \
-S ldap -K profiles="Network Management Plus" netmgtsec
# passwd netmgtsec
New Password: xxxxxxxx
Confirm password: xxxxxxxx
# usermod -R netmgtsec jdoe
```

The rights in the profiles are available to the user `jdoe` after `jdoe` assumes the `netmgtsec` role.

```
$ su - netmgtsec
Password: xxxxxxxx
$
```

**Example 26** Dividing Network Security Responsibilities Between Roles

In this example, the administrator divides network security responsibilities between two roles. One role administers Wifi and link security and another role administers IPsec and IKE. Each role is assigned to three people, one person per shift.

The roles are created by the administrator as follows:

1. The administrator names the first role LinkWifi.
2. The administrator assigns the Network Wifi, Network Link Security, and Network Management rights profiles to the role.
3. The administrator assigns the LinkWifi role to the appropriate users.
4. The administrator names the second role IPsec Administrator.
5. The administrator assigns the Network IPsec Management and the Network Management rights profiles to the role.
6. The administrator assigns the IPsec Administrator role to the appropriate users.

**Example 27** Enabling a Trusted User to Configure and Manage IPsec

In this example, the administrator gives one user responsibility for configuring and managing IPsec.

In addition to the Network Management and IPsec Network Management rights profiles, the administrator gives the user the ability to edit the `hosts` file and the ability to read the logs.

1. The administrator creates two rights profiles, one for editing files and the other for reading logs.

   ```
   $ profiles -p -S LDAP "Hosts Configuration"
   profiles:Network Configuration> set desc="Edits root-owned network files"
   ...Configuration> add auth=solaris.admin.edit/etc/hosts
   ...Configuration> commit
   ...Configuration> end
   ```

```
...Configuration> exit

# profiles -p -S LDAP "Read Network Logs"
profiles:Read Network Logs> set desc="Reads root-owned network log files"
...Logs> add cmd=/usr/bin/more
...Logs:more>set privs={file_dac_read}:/var/user/ikeuser/*
...Logs:more>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:more> set privs={file_dac_read}:/etc/inet/ike/*
...Logs:more> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:more>end
...Logs> add cmd=/usr/bin/tail
...Logs:tail>set privs={file_dac_read}:/var/user/ikeuser/*
...Logs:tail>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:tail>set privs={file_dac_read}:/etc/inet/ike/*
...Logs:tail> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:tail>end
...Logs> add cmd=/usr/bin/page
...Logs:page>set privs={file_dac_read}:/var/user/ikeuser/*
...Logs:page>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:page>set privs={file_dac_read}:/etc/inet/ike/*
...Logs:page> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:page>end
...Logs> exit
```

The rights profile enables the user to use the `more`, `tail`, and `page` commands to read the logs. The `cat` and `head` commands cannot be used.

2. The administrator creates the rights profile that enables the user to perform all configuration and management tasks for IPsec and its keying services.

```
# profiles -p "Site Network Management"
profiles:Site Network Management> set desc="Handles all network files and logs"
...Management> add profiles="Network Management"
...Management> add profiles="Network IPsec Management"
...Management> add profiles="Hosts Configuration"
...Management> add profiles="Read Network Logs"
...Management> commit; end; exit
```

3. The administrator creates a role for the profile, assigns it a password, and assigns the role to a trusted user who understands networking and security.

```
# roleadd -S LDAP -c "Network Management Guru" \
-m -K profiles="Site Network Management" netadm
# passwd netadm
Password: xxxxxxxx
Confirm password: xxxxxxxx
```

```
# usermod -S LDAP -R +netadm jdoe
```

4. Out of band, the administrator supplies `jdoe` with the role password.

## ▼ How to Verify That Packets Are Protected With IPsec

To verify that packets are protected, test the connection with the `snoop` command. The following prefixes can appear in the `snoop` output:

- `AH:` Prefix indicates that AH is protecting the headers. You see this prefix if you used `auth_alg` to protect the traffic.
- `ESP:` Prefix indicates that encrypted data is being sent. You see this prefix if you used `encr_auth_alg` or `encr_alg` to protect the traffic.

**Before You Begin** You must have access to both systems to test the connection.

You must assume the `root` role to create the `snoop` output. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **On one system, such as `host2`, assume the `root` role.**

```
$ su -
Password: xxxxxxxx
#
```

2. **(Optional) Display the details of the SAs.**

```
# ipseckey dump
```

This output indicates which SPI values match the SAs that are used, which algorithms were used, the keys, and so on.

3. **On this system, prepare to snoop packets from a remote system.**
In a terminal window on `host2`, snoop the packets from the `host1` system.

```
# snoop -d net0 -o /tmp/snoop_capture host1
Using device /dev/xxx (promiscuous mode)
```

4. **Send a packet from the remote system.**
In another terminal window, remotely log in to the `host1` system. Provide your password. Then, assume the `root` role and send a packet from the `host1` system to the `host2` system. The packet should be captured by the `snoop -v host1` command.

```
host2 $ ssh host1
Password: xxxxxxxx
host1 $ su -
Password: xxxxxxxx
host1 $ ping host2
```

5. **Examine the `snoop` output.**

```
host2 # snoop -i /tmp.snoop_capture -v
```

You can also load the `snoop` output into the Wireshark application. For more information, see "How to Prepare IPsec and IKE Systems for Troubleshooting" on page 198 and "`snoop` Command and IPsec" on page 221.

In the file, you should see output that includes `AH` and `ESP` information after the initial `IP` header information. `AH` and `ESP` information that resembles the following shows that packets are being protected:

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 4e0e
IP:   Source address = 198.51.100.6, host1
IP:   Destination address = 198.51.100.33 host2
IP:   No options
IP:
AH:  ----- Authentication Header -----
AH:
AH:  Next header = 50 (ESP)
AH:  AH length = 4 (24 bytes)
AH:  <Reserved field = 0x0>
AH:  SPI = 0xb3a8d714
AH:  Replay = 52
AH:  ICV = c653901433ef5a7d77c76eaa
AH:
ESP:  ----- Encapsulating Security Payload -----
ESP:
ESP:  SPI = 0xd4f40a61
ESP:  Replay = 52
ESP:     ....ENCRYPTED DATA....

ETHER:  ----- Ether Header -----
...
```

♦ ♦ ♦ CHAPTER 8

# About Internet Key Exchange

Internet Key Exchange (IKE) automates key management for IPsec. This chapter contains the following information about IKE:

- "Introduction to IKE" on page 129
- "IKEv2 Protocol" on page 135
- "IKEv1 Protocol" on page 138

For instructions on implementing the latest version of the IKE protocol, see Chapter 9, "Configuring IKEv2". To continue to use IKEv1, see Chapter 10, "Configuring IKEv1". For reference information, see Chapter 12, "IPsec and Key Management Reference". For information about IPsec, see Chapter 6, "About IP Security Architecture".

## Introduction to IKE

The management of keying material for IPsec security associations (SAs) is called *key management*. Automatic key management requires a secure channel of communication for the creation, authentication, and exchange of keys. Oracle Solaris uses Internet Key Exchange (IKE) to automate key management. IKE eliminates administrative overhead and the security risk of manually distributing secret keys.

Oracle Solaris supports two versions of the IKE protocol.

- IKE Version 2 (IKEv2), which is based on Internet Key Exchange Protocol Version 2 (IKEv2), RFC 5996
- IKE Version 1 (IKEv1), which is based on The Internet Key Exchange (IKE), RFC 2409

On a FIPS 140-2 enabled system, you should configure IKEv2 with FIPS 140-2 approved algorithms only. For more information, see "IKEv2 and FIPS 140-2" on page 137.

---

**Note -** If you have a strict requirement to use only FIPS 140-2 validated cryptography, you must be running the Oracle Solaris 11.3 SRU 5.6 release. Oracle completed a FIPS 140-2 validation against the Cryptographic Framework in this specific release. Oracle Solaris 11.4 builds on this validated foundation and includes software improvements that address performance, functionality, and reliability. Whenever possible, you should configure Oracle Solaris 11.4 in FIPS 140-2 mode to take advantage of these improvements.

---

## IKE Concepts and Terminology

The following concepts and terms are common to both versions of IKE. They might be implemented differently in the two versions.

- **Key negotiation and exchange –** The exchange of keying material and the authentication of the peer's identity in a secure manner. The process uses asymmetric cryptographic algorithms. The two main methods are the RSA and the Diffie-Hellman protocols.

  IKE creates and manages the IPsec SAs between systems that are running an IKE daemon. IKE negotiates a secure channel that protects the transmission of keying material. The daemon creates the keys from a random number generator by using the `/dev/random` device. The daemon changes the keys at a configurable rate. The keying material is available to algorithms that are specified in the configuration file for IPsec policy, `ipsecinit.conf`.
- **Diffie-Hellman (DH) algorithm –** A key exchange algorithm that allows two systems to securely generate a shared secret over an insecure channel.
- **RSA algorithm –** An asymmetric key algorithm that is used to authenticate the identity of peer systems, typically by proving ownership of an X.509 certificate. The algorithm is named for its three creators: Rivest, Shamir, and Adleman.

  Alternatively, DSA or ECDSA algorithms may be used for this purpose.
- **Perfect forward secrecy (PFS) –** In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys. Therefore, PFS can prevent the decryption of previously recorded traffic.
- **Oakley group –** Used to negotiate PFS. See Section 6 of The Internet Key Exchange (IKE).
- **IKE policy –** The set of IKE rules which define the acceptable parameters that an IKE daemon uses when attempting to set up a secure key exchange channel with a peer system. This is called an IKE SA in IKEv2 or Phase 1 in IKEv1.

  The parameters include algorithms, key sizes, Oakley groups, and authentication method. The Oracle Solaris IKE daemons support preshared keys and certificates as authentication methods.

## How IKE Works

A system that is running an IKE daemon can negotiate the parameters needed to create a security association (SA) between this system and another system that is running the IKE daemon. The protocol that is used to negotiate this SA and subsequent IPsec SAs is known as IKE. This version of Oracle Solaris supports version 1 (IKEv1) and version 2 (IKEv2) of the IKE protocol.

The IKE security association (also known as the ISAKMP or Phase 1 SA in IKEv1) secures further protocol exchanges between these two IKE systems. These exchanges negotiate cryptographic algorithms, IPsec policy, and other parameters needed to create IPsec SAs.

Systems that are running an IKE daemon can also be configured to negotiate IPsec SAs on behalf of other systems. When configured in this manner, the systems are referred to as *security gateways*. If the IKE negotiation is successful, the IPsec SAs can be used to protect network packets.

The parameters that are negotiated to create the IKE SA include the cryptographic algorithms that protect the IKE exchanges and some authentication material. The authentication material is used to determine whether the packets that contain the IKE protocol exchanges can be trusted. Trust means that the packets come from a trusted system and not from a system that is pretending to be that system.

Oracle Solaris supports two types of authentication material for IKE, preshared keys and public key certificates.

### IKE With Preshared Key Authentication

A preshared key is string of hex or ASCII characters that only the two IKE systems know. The keys are called *preshared* because both endpoints must know the value of the key before the IKE exchange. This key must be part of the IKE configuration on both systems. The preshared key is used in the generation of the IKE payloads, which make up the packets that implement the IKE protocol. The system that processes these IKE payloads uses the same key to authenticate the payloads that it receives.

The preshared key is not exchanged between the IKE endpoints by using the IKE protocol. Typically, the key is shared with the peer system over a different medium, such as a phone call.

The preshared key on the peers that use this authentication method must be identical. The keys are stored in a file on each system.

## IKE With Public Key Certificates

Public key certificates and their trust chains provide a mechanism to digitally identify a system without having to manually exchange any secret information. Therefore, public key certificates are more secure than preshared keys.

A public key certificate is a blob of data that encodes a public key value, some information about the generation of the certificate, such as a name and who signed it, a hash or checksum of the certificate, and a digital signature of the hash. Together, these values form the certificate. The digital signature ensures that the certificate has not been modified.

A public key is a value that is mathematically derived from another value, called the *private key*. The mathematical algorithm that derives the public key from the private key makes retrieving the private key from the public key impractical. Therefore, public key certificates can be freely shared. Examples of algorithms that are used to derive public keys include RSA and Elliptic Curve.

A digital signature is the result of passing the certificate contents through a digital signing algorithm such as RSA, DSA or ECDSA. These algorithms use a private signing key, which is not part of the certificate, and produce a digital signature. The signature is appended to the certificate. Again, calculating the signing key from the certificate contents and the signature is impractical. More to the point, the certificate signature and hence the certificate contents can be easily verified by using a public value, which was derived from the signing key.

A certificate can be self-signed, in which case the signature of the certificate can be verified by the certificate's public key, or it can be signed by a different entity. When a different entity signs the certificate, the public key value that is used to verify the certificate is also distributed as a public key certificate. This second certificate will be signed by a certificate authority (CA) which is trusted, or by an intermediary. The intermediary is ultimately trusted by the signing entity, that is, the root CA.

These public key certificate components, plus the procedures and structures that implement them are often referred to as a public key infrastructure (PKI). The scope of an organization's PKI can vary. A simple PKI could consist of a CA that signs a few certificates for local use. A more extensive PKI would use a globally recognized trust anchor as the authoritative CA.

### Using Public Key Certificates in IKE

This section describes the overall steps to create and use public key certificates in IKE. For the specific procedures, see “Configuring IKEv2 With Preshared Keys” on page 142 and “Configuring IKEv1 With Preshared Keys” on page 168.

1. To use either a self-signed certificate or a certificate from a certificate authority (CA), you first generate a public/private key pair.
   - For a self-signed certificate, IKE peers then exchange these certificates, verify out of band that the certificates are genuine, and then import the peers' certificates into the local keystore. The keystore then contains the original self-signed certificate plus the imported certificates.
   - For certificates from a CA, you perform several more steps. When you generate the public/private key pair, you also generate a certificate signing request (CSR). A CSR contains the public key and *identifiers*. A typical identifier is a distinguished name (DN), for example:

     ```
     DN=O=Example\, Inc, OU=qa, L=Silicon Valley, ST=CA, CN=host1
     ```

     ---

     **Tip -** Create a DN or other identifier that is as specific as possible to reduce the possibility of matching another certificate's identifier.

     ---
2. Send the CSR to the CA for signature.

   In a typical process, you paste the CSR into a web form and submit the form to the CA. The CA might send more than one signed certificate to you.
3. Get the signed certificates from the CA, then import them into your IKEv2 keystore or IKEv1 database.

   You must import all the certificates that the CA sends. These certificates comprise a "chain of trust" from the trust anchor, or root CA, to your individually identified signed certificate.
4. Repeat the process on an IKE peer.
5. Use the certificates in IKE rules.

   You specify the certificate by an identifier, such as a DN. For CA-signed certificates, you can configure IKE to accept any certificate that is signed by a particular CA.

## Handling Revoked Certificates

A signed certificate is trusted as valid because the signing authority assures its validity. If a certificate is compromised or otherwise determined as invalid, the CA will revoke it.

CAs maintain a list of revoked certificates, often called the certificate revocation list (CRL). You can use the Online Certificate Status Protocol (OCSP) to dynamically check the status of a certificate. Some public key certificates have URIs embedded in them. They identify a web location where you can check the CRL or the web location of an OCSP server.

For more information, see Section 3.3 of Internet X.509 Public Key Infrastructure Certificate and CRL Profile, RFC 2459 and the RFC X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP, RFC 2560.

### Coordinating Time on Systems That Use Public Certificates

Public key certificates contain the date and time of issue and the time that they remain valid. Therefore, the clocks on systems that generate and use certificates must be accurate. The Network Time Protocol (NTP) software can be used to synchronize the clocks on systems. NTP public domain software from the University of Delaware is included in the Oracle Solaris release. Documentation is available from the NTP Documentation web site. You can also install the `service/network/ptp` package to configure the Precision Time Protocol (PTP) service. See IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (`https://www.nist.gov/el/intelligent-systems-division-73500/ieee-1588`).

## Specifying an IKE Version

You can require the use of a specific IKE protocol version on a per-rule basis, which enables you to change only the IPsec rule while not changing its corresponding IKE configuration rule. Specifying the IKEv2 protocol in an IPsec rule is a useful step when transitioning systems in an orderly fashion from the legacy IKEv1 protocol. For an example, see Example 21, “Configuring IPsec Policy to Use the IKEv2 Protocol Only,” on page 108.

## Comparison of IKEv2 and IKEv1

The following table compares the implementation of the IKEv2 and IKEv1 versions on an Oracle Solaris system.

**TABLE 9** IKEv2 and IKEv1 Implementation in Oracle Solaris

| Configuration Component | IKEv2 | IKEv1 |
|---|---|---|
| Certificate chain of trust | Implicit based on objects in keystore | `cert_trust` parameter in `ike/config` file |
| Certificate creation | `ikev2cert` command | `ikecert certlocal` command |
| Certificate import | `ikev2cert import` command can import certificates and keys into PKCS #11 keystore | `ikecert certdb` command can import standalone certificates into IKE keystore |

| Configuration Component | IKEv2 | IKEv1 |
|---|---|---|
| Certificate owner | `ikeuser` | `root` |
| Certificate policy file | `kmf-policy.xml` | Some policy in `ike/config` file |
| Certificate storage | PKCS #11 softtoken library | Local IKEv1 databases |
| Configuration file directory | `/etc/inet/ike/` | `/etc/inet/ike/` and `/etc/inet/secret/` |
| Configuration owner | `ikeuser` account | `root` account |
| Daemon | `in.ikev2d` | `in.iked` |
| IKE message length | Size can be adjusted to path MTU | Size cannot be adjusted |
| IKE policy file | `ike/ikev2.config` | `ike/config` |
| IKE preshared keys | `ike/ikev2.preshared` | `secret/ike.preshared` |
| IKE SAs and protocol exchanges† | Can be configured with FIPS 140-2 approved algorithms | Cannot be configured with FIPS 140-2 approved algorithms |
| Certificate operations† | Can be configured with FIPS 140-2 approved algorithms | Cannot be configured with FIPS 140-2 approved algorithms |
| NAT port | UDP port `4500` | UDP port `4500` |
| Port | UDP port `500` | UDP port `500` |
| Rights profile | Network IPsec Management | Network IPsec Management |
| Service name (FMRI) | `svc:/ipsec/ike:ikev2` | `svc:/ipsec/ike:default` |

†The Cryptographic Framework feature of Oracle Solaris 11.3 SRU 5.6 is validated for FIPS 140-2 Level 1. If FIPS 140-2 mode is enabled and the Cryptographic Framework is being used, then FIPS 140-2 approved algorithms are available. By default, FIPS 140-2 mode is not enabled.

## IKEv2 Protocol

This section covers the implementation of IKEv2. For IKEv1 information, see "IKEv1 Protocol" on page 138. For a comparison, see "Comparison of IKEv2 and IKEv1" on page 134. For information that applies to both protocols, see "Introduction to IKE" on page 129. Oracle Solaris supports both versions of the IKE protocol simultaneously.

The IKEv2 daemon, `in.ikev2d`, negotiates and authenticates keying material for IPsec SAs. See the `in.ikev2d(8)` man page.

### IKEv2 Configuration Choices

The `/etc/inet/ike/ikev2.config` configuration file contains the configuration for the `in.ikev2d` daemon. The configuration consists of a number of rules. Each entry contains parameters such as algorithms and authentication data that this system can use with a similarly configured IKEv2 peer.

The `in.ikev2d` daemon supports preshared keys (PSK) and public key certificates for identity.

The `ikev2.config`(5) man page provides sample rules. Each rule must have a unique label. The following is a list of the descriptive labels of sample rules from the man page:

- `IP identities and PSK auth`
- `IP address prefixes and PSK auth`
- `IPv6 address prefixes and PSK auth`
- `Certificate auth with DN identities`
- `Certificate auth with many peer ID types`
- `Certificate auth with wildcard peer IDs`
- `Override transforms`
- `Mixed auth types`
- `Wildcard with required signer`

---

**Note -** A preshared key can be used with any one of many peer ID types, including IP addresses, DNs, FQDNs, and email addresses.

---

## IKEv2 Policy for Public Certificates

The `kmf-policy.xml` file contains the certificate validation policy for IKEv2. The `kmfcfg dbfile=/etc/inet/ike/kmf-policy.xml policy=default` command is used to modify certificate validation policy. Typical modifications include the use of OCSP and CRLs, and the duration of network timeouts during certificate verification. Additionally, the policy enables an administrator to modify various aspects of certificate validation, such as validity date enforcement and key usage requirements. Loosening the default requirements for certificate validation is not recommended.

## IKEv2 Messages Across Intermediate Devices

When IP packets carrying IKE messages are large, the packets can become fragmented at the IP layer. Some intermediate devices, such as NAT boxes and firewalls, drop IP fragments, which results in the loss of IKE messages and timeouts. In IKEv2, you can limit the IP packets to a length below the IP fragmentation threshold. The default IKEv2 fragmentation threshold is 1350 bytes. For more information, see "IKEv2 Service" on page 224 and Example 44, "Preventing the Loss of IKEv2 Messages From Intermediate Devices," on page 204.

## IKEv2 and FIPS 140-2

On a FIPS 140-2 enabled system, you are responsible for choosing only FIPS 140-2 approved algorithms when creating certificates and configuring IKEv2. The procedures and examples in this guide use FIPS 140-2 approved algorithms except when the algorithm "`any`" is specified.

The following encryption algorithm mechanisms are available to use in the IKEv2 configuration and preshared keys files and approved for use in Oracle Solaris in FIPS 140-2 mode:

- AES in CBC mode in 128-bit to 256-bit key lengths
- 3DES

The following authentication algorithm mechanisms are available to use in IKEv2 configuration and preshared keys files and approved for use in Oracle Solaris in FIPS 140-2 mode:

- SHA1
- SHA256
- SHA384
- SHA512

The following mechanisms are available to use in IKEv2 certificates and approved for use in Oracle Solaris in FIPS 140-2 mode:

- RSA in 2048-bit to 3072-bit key lengths
- ECDSA that uses ECC with three possible curves and their associated hashes –

  The arguments to the `ikev2cert gencert` and `ikev2cert gencsr` commands are the following:

  - `keytype=ec curve=secp256r1 hash=sha256`
  - `keytype=ec curve= secp384r1 hash=sha384`
  - `keytype=ec curve=secp521r1 hash=sha512`

  For more information, see the `ikev2cert(8)` man page.

For the definitive list of FIPS 140-2 approved algorithms for Oracle Solaris, follow the links in "FIPS 140-2 Level 1 Guidance Documents for Oracle Solaris Systems" in *Using a FIPS 140-2 Enabled System in Oracle Solaris 11.4*.

## IKEv1 Protocol

The following sections provide an overview of IKEv1. IKEv1 is superseded by IKEv2, which offers faster, secured key management. For information about IKEv2, see "IKEv2 Protocol" on page 135. For a comparison, see "Comparison of IKEv2 and IKEv1" on page 134. For information that is common to both protocols, see "Introduction to IKE" on page 129. IKEv1 and IKEv2 can run simultaneously and negotiate with their peer protocol on other systems.

## IKEv1 Key Negotiation

The IKEv1 daemon, `in.iked`, negotiates keys and authenticates IPsec SAs in a secure manner. IKEv1 provides perfect forward secrecy (PFS). In PFS, the keys that protect data transmission are not used to derive additional keys. Also, seeds used to create data transmission keys are not reused. See the `in.iked(8)` man page.

### IKEv1 Phase 1 Exchange

The IKEv1 protocol has two phases. Oracle Solaris supports the *Main Mode* Phase 1 exchange. The Main Mode exchange negotiates acceptable parameters to create an ISAKMP security association (SA) between the two peers. This ISAKMP SA uses asymmetrical encryption to exchange its keying material and authenticates its peer using a preshared key or a public key certificate. Unlike IPsec SAs, the ISAKMP SAs are bidirectional, so only one security association is needed.

How IKEv1 negotiates ISAKAMP SAs in the Phase 1 exchange is configurable. IKEv1 reads the configuration information from the `/etc/inet/ike/config` file. Configuration information includes the following:

- Global parameters, such as the names of public key certificates
- Whether perfect forward secrecy (PFS) is required
- This system's IKE peers
- The algorithms that protect Phase 1 exchanges
- The authentication method

  The two authentication methods are preshared keys and public key certificates. The public key certificates can be self-signed or issued by a certificate authority (CA).

For more information, see the `ike.config(5)` man page.

### IKEv1 Phase 2 Exchange

The Phase 2 exchange is known as *Quick Mode*. The Quick Mode exchange negotiates the IPsec algorithms and keying material that is needed to create IPsec SAs. This exchange is protected (encrypted) by the ISAKMP SA that is negotiated in Phase 1.

The algorithms and security protocols in the Quick Mode exchange come from the IPsec policy file, `/etc/inet/ipsecinit.conf`.

The IPsec SAs are rekeyed when they expire. The lifetime of the SA is set by the `in.iked` daemon when it creates the IPsec SA. This value is configurable.

For more information, see the `ipsecconf`(8) and `in.iked`(8) man pages.

## IKEv1 Configuration Choices

The `/etc/inet/ike/config` configuration file contains the configuration for the `in.iked` daemon. The configuration consists of a number of rules. Each entry contains parameters such as algorithms and authentication data that this system can use with a similarly configured IKEv1 peer. The `in.iked` daemon supports preshared keys and public key certificates for identity.

The entry `auth_method preshared` indicates that preshared keys are used. Values for `auth_method` other than `preshared` indicate that public key certificates are to be used.

In IKEv1, preshared keys are tied to a particular IP address or range of addresses. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system.

For more information, see "How IKE Works" on page 131 and the `ike.config`(5) and `ike.preshared`(5) man pages.

♦ ♦ ♦ CHAPTER 9

# Configuring IKEv2

This chapter describes how to configure the Internet Key Exchange version 2 (IKEv2) for your systems. After IKEv2 is configured and enabled, it automatically generates keying material for the IPsec endpoints that it specifies. This chapter contains the following information:

- "Configuring IKEv2" on page 141
- "Configuring IKEv2 With Preshared Keys" on page 142
- "Initializing the Keystore to Store Public Key Certificates for IKEv2" on page 149
- "Configuring IKEv2 With Public Key Certificates" on page 152

For overview information about IKE, see Chapter 8, "About Internet Key Exchange". For reference information about IKE, see Chapter 12, "IPsec and Key Management Reference". For more procedures, see the examples in the `ikeadm`(8), `pktool`(1), `ikev2cert`(8), `ikev2.config`(5), `in.ikev2d`(8), and `kmfcfg`(1) man pages.

## Configuring IKEv2

You can use preshared keys, self-signed certificates, and certificates from a certificate authority (CA) to authenticate IKE. Rules link a particular authentication method with the end points that are being protected. Therefore, you can use one or all authentication methods on a system. You can also run IKEv1 on an IKEv2 system. Typically, you run IKEv1 to protect communications with systems that do not support IKEv2.

After configuring IKEv2, complete the IPsec procedures in Chapter 7, "Configuring IPsec" that use these IKEv2 rules to manage their keys. The following sections focus on specific IKEv2 configurations.

## Configuring IKEv2 With Preshared Keys

If you are configuring peer systems or subnets to use IKEv2, and you are the administrator of these subnets, using preshared keys can be a good choice. Preshared keys might also be used when testing. For more information, see "IKE With Preshared Key Authentication" on page 131.

### ▼ How to Configure IKEv2 With Preshared Keys

Substitute the names of your systems for the names `host1` and `host2` in this procedure. You configure both IKE endpoints.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **On each system, edit the `/etc/inet/ike/ikev2.config` file.**

```
# pfedit /etc/inet/ike/ikev2.config
```

2. **In the file, create a rule that uses preshared keys.**

---

**Note -** You will create the keys in Step 5.

---

The rules and global parameters in this file must manage the keys in the IPsec policy in the system's `ipsecinit.conf` file. The following IKEv2 configuration examples manage the keys of the `ipsecinit.conf` examples in "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

a. **For example, modify the `ikev2.config` file on the `host1` system:**

---

**Note -** This example shows two transforms in the global parameters section. A peer can be configured with either of these transforms. To require a particular transform, include that transform in the rule.

---

```
### ikev2.config file on host1, 192.0.2.16

## Global parameters
# This default value will apply to all transforms that follow
#
ikesa_lifetime_secs 3600
#
# Global transform definitions.  The algorithm choices are
# based on RFC 4921.
#
## Two transforms are acceptable to this system, Group 20 and Group 19.
## A peer can be configured with 19 or 20.
## To ensure that a particular peer uses a specific transform,
## include the transform in the rule.
##
# Group 20 is 384-bit ECP - Elliptic Curve over Prime
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
# Group 19 is 256-bit ECP
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
#
## The rule to communicate with host2
##  Label must be unique
{ label "host1-host2"
  auth_method preshared
  local_addr  192.0.2.16
  remote_addr 192.0.2.213
}
```

b. **Modify the `ikev2.config` file on the `host2` system:**

```
## ikev2.config file on host2, 192.0.2.213
## Global Parameters
#
...
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
...
## The rule to communicate with host1
##  Label must be unique
{ label "host2-host1"
  auth_method preshared
  local_addr  192.0.2.213
  remote_addr 192.0.2.16
}
```

3. **On each system, verify the syntax of the file.**

```
# /usr/lib/inet/in.ikev2d -c
```

4. **Create a preshared key for IKEv2 to use.**

   An AES key of at least 256 bits is a good choice.

   - For a full description of how to create a key, see "How to Generate a Symmetric Key by Using the pktool Command" in *Managing Encryption and Certificates in Oracle Solaris 11.4*.
   - For examples of key generation, see Example 28, "Generating a Preshared Key for IKEv2," on page 145 and Example 29, "Using Different Local and Remote IKEv2 Preshared Keys," on page 145.

5. **Put the preshared key in the `/etc/inet/ike/ikev2.preshared` file on each system.**

   **Caution -** This file has special permissions and is owned by `ikeuser`. Never delete or replace this file. Instead, use the `pfedit -s` command to edit its contents so that the file retains its original properties and the contents do not appear in the audit log.

   a. **For example, on the `host1` system, the `ikev2.preshared` file would appear similar to the following:**

   ```
   # pfedit -s /etc/inet/ike/ikev2.preshared
   ## ikev2.preshared on host1, 192.0.2.16
   # ...
   ## label must match the rule that uses this key
   { label "host1-host2"
      key  "1011e1f2d1fd..."
   }
   ```

   For information about the options to the `pfedit` command, see the `pfedit(8)` man page.

   b. **On the `host2` system, the `ikev2.preshared` file is similar except for its unique label:**

   ```
   ## ikev2.preshared on host2, 192.0.2.213
   # ...
   ## label must match the label of the rule that uses this key
   { label "host2-host1"
      key  "1011e1f2d1fd..."
   }
   ```

6. **Enable the IKEv2 service instance.**

   ```
   # svcadm enable ipsec/ike:ikev2
   ```

When replacing the preshared key, edit the preshared key files on the peer systems and restart the `ikev2` service.

```
# svcadm restart ikev2
```

**Example 28** Generating a Preshared Key for IKEv2

In the following example, the administrator manually creates the keying material for two systems that are protected by IKE, `local1` and `remote1`. The label of the preshared key entry matches the label in a rule in the `ikev2.config` file. Then, the administrator copies the key to the `/etc/ike/ikev2.preshared` file and destroys the original key file.

1. First, the administrator creates and displays the preshared key.

   ```
   local1$ pktool genkey keystore=file outkey=ike2psk keytype=aes keylen=256 print=y
   Key Value ="2b823670b5aa1a..."
   ```
2. The administrator adds the key to the `ikev2.preshared` file on `local1`.

   ```
   { label "local1-remote1"
      key  "2b823670b5aa1a..."
   }
   ```
3. The administrator destroys the original key file.

   ```
   $ rm ike2psk
   ```
4. The administrator copies the `ikev2.preshared` file to the communicating system by using the `ssh` command or another secure mechanism.
5. On `remote1`, the administrator appends the `ikev2.preshared` file to an existing `/etc/inet/ike/ikev2.preshared` file, or creates a new file.

   Then, the administrator changes the label to match the rule in the `ikev2.config` file.

   ```
   { label "remote1-local1"
      key "2b823670b5aa1a..."
   }
   ```

**Example 29** Using Different Local and Remote IKEv2 Preshared Keys

In this example, the IKEv2 administrators create a preshared key per system, exchange them, and add each key to the `/etc/inet/ike/ikev2.preshared` file. The label of the preshared key entry matches the label in a rule in the `ikev2.config` file. Then, they restart the `in.ikev2d` daemons.

1. On `host1`, the administrator generates two keys.

```
$ pktool genkey keystore=file outkey=ikemykey keytype=aes keylen=256 print=y
Key Value ="e6fc5402efd08..."
$ pktool genkey keystore=file outkey=ikeotherkey keytype=aes keylen=256 print=y
Key Value ="01ca0f4d32923..."
```

2. The administrator places the keys in the `ikev2.preshared` file.

```
##...
{ label "host1-host2"
## local and remote preshared keys
  local_key  "e6fc5402efd08..."
  remote_key "01ca0f4d32923..."
}
```

3. The administrator destroys the original keys files.

```
$ rm ikemykey ikeotherkey
```

4. The administrator copies the `ikev2.preshared` file to `host2` by using the `ssh` command or another secure mechanism.
5. After receiving the other system's preshared key, the administrator edits the `ikev2.preshared` file. The file on `host2` is the following:

```
##...
{ label "host2-host1"
## local and remote preshared keys
  local_key  "01ca0f4d32923..."
  remote_key "e6fc5402efd08..."
}
```

6. The administrators restart the IKEv2 service instance on each system.

```
# svcadm restart ikev2
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

For more examples, see the `ikev2.config(5)` and `ikev2.preshared(5)` man pages.

## ▼ How to Add a New Peer When Using Preshared Keys in IKEv2

If you add IPsec policy entries to a working configuration between the same peers, you need to refresh the IPsec policy service. You do not need to reconfigure or restart IKE.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKEv2 configuration.

**Before You Begin** You have updated the `ipsecinit.conf` file and refreshed IPsec policy for the peer systems.

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Create a rule for IKEv2 to manage the keys for the new system that is using IPsec.**

   a. **For example, on the `host1` system, add the following rule to the `/etc/inet/ike/ikev2.config` file:**

   ```
   # pfedit ikev2.config
   ## ikev2.config file on host1, 192.0.2.16
   ...
   ## The rule to communicate with host3
   ##  Label must be unique
   {label "host1-host3"
    auth_method preshared
    local_addr  192.0.2.16
    remote_addr 192.0.2.7
   }
   ```

   For information about the options to the `pfedit` command, see the `pfedit(8)` man page.

   b. **On the `host3` system, add the following rule:**

   ```
   ## ikev2.config file on host3, 192.0.2.7
   ...
   ## The rule to communicate with host1
   {label "host3-host1"
   ```

```
 auth_method preshared
 local_addr  192.0.2.7
 remote_addr 192.0.2.16
}
```

2. **(Optional) On each system, verify the syntax of the file.**

   ```
   # /usr/lib/inet/in.ikev2d -c -f /etc/inet/ike/ikev2.config
   ```

3. **Create an IKEv2 preshared key for the peer systems.**

   a. **Generate the key on `host1`.**

   ```
   $ pktool genkey keystore=file outkey=ikemykey keytype=aes keylen=256 print=y
   Key Value ="2b823670b5aa1a..."
   ```

   b. **On the `host1` system, add the following information to the `/etc/inet/ike/ikev2.preshared` file:**

   ```
   # pfedit -s /etc/inet/ike/ikev2.preshared
   ## ikev2.preshared on host1 for the host3 interface
   ...
   ## The rule to communicate with host3
   ##  Label must match the label of the rule
   { label "host1-host3"
     # host1 and host3's shared key
       key "2b823670b5aa1a..."
   }
   ```

   For information about the options to the `pfedit` command, see the `pfedit`(1M) man page.

   c. **Remove the key file from `host1` and send the `ikev2.preshared` file to `host3` by a secure mechanism.**

   d. **On the `host3` system, add the following information to the `ikev2.preshared` file:**

   ```
   ## ikev2.preshared on host3 for the host1 interface
   # ...
   { label "host3-host1"
     # host3 and host1's shared key
       key "2b823670b5aa1a..."
   }
   ```

4. **On each system, read the changes into the kernel.**

- **If the service is enabled, refresh it.**

  ```
  # svcadm refresh ikev2
  ```

- **If the service is not enabled, enable it.**

  ```
  # svcadm enable ikev2
  ```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## Initializing the Keystore to Store Public Key Certificates for IKEv2

To use public certificates with IKEv2, you must create a PKCS #11 keystore. The most commonly used keystore uses `pkcs11_softtoken`, which is provided by the Cryptographic Framework feature of Oracle Solaris.

The `pkcs11_softtoken` keystore for IKEv2 is in a directory that is owned by a special user, `ikeuser`. The default directory is `/var/user/ikeuser`. The user ID `ikeuser` is delivered with the system, but you must create the keystore. When you create the keystore, you create a PIN for the keystore. The IKEv2 service requires this PIN to log in to the keystore.

The `pkcs11_softtoken` keystore holds the private keys, public keys, and public certificates that are used by IKEv2. These keys and certificates are managed with the `ikev2cert` command, which is a wrapper for the `pktool` command. The wrapper ensures that all keys and certificate operations are applied to the `pkcs11_softtoken` keystore that is owned by `ikeuser`.

If you have not added the PIN as a property value of the `ikev2` service, the following message displays in the `/var/log/ikev2/in.ikev2d.log` file:

```
date: (n)  No PKCS#11 token "pin" property defined
for the smf(5) service: ike:ikev2
```

If you are not using public key certificates, you can ignore this message.

## ▼ How to Create and Use a Keystore for IKEv2 Public Key Certificates

You must create a keystore if you plan to use public certificates with IKEv2. To use the keystore, you must log in to it. When the `in.ikev2d` daemon starts, you or an automatic process supplies the PIN to the daemon. If site security permits automatic login, you must configure it. The default is an interactive login to use the keystore.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Set the PIN for the IKEv2 keystore.**

   You use the `ikev2cert setpin` command to create the IKEv2 keystore. This command sets the owner of the PKCS #11 keystore to be `ikeuser`.

   Do not use spaces in the PIN. For example, the value `WhatShouldIWrite` is valid, but the value `"What Should"` is not.

   ```
   $ pfbash
   # /usr/sbin/ikev2cert setpin
   Enter token passphrase: changeme
   Create new passphrase:     Type strong passphrase
   Re-enter new passphrase: xxxxxxxx
   Passphrase changed.
   ```

   **Caution -** Store this passphrase in a safe location. You need it to use the keystore.

2. **Log in to the keystore automatically or interactively.**

   Automatic login is preferred. If site security policy does not permit automatic login, you must interactively log in to the keystore when the `in.ikev2d` daemon is restarted.

   - **Configure the keystore to enable automatic login.**

     a. **Add the PIN as the value for the `pkcs11_softtoken/pin` service property.**

     ```
     # svccfg -s ike:ikev2 editprop
     ```

     A temporary edit window opens.

     b. **Uncomment the `setprop pkcs11_token/pin =` line.**

```
# setprop pkcs11_token/pin = astring: ()       Original entry
setprop pkcs11_token/pin = astring: () Uncommented entry
```

c. **Replace the parentheses with the PIN from Step 1.**

```
setprop pkcs11_token/pin = astring: PIN-from-Step-1
```

Leave a space between the colon and the PIN.

d. **Uncomment the `refresh` line at the bottom of the file, then save your changes.**

```
# refresh
refresh
```

e. **(Optional) Verify the value of the `pkcs11_token/pin` property.**

The `pkcs11_token/pin` property holds the value that is checked when accessing the keystore owned by `ikeuser`.

```
# svccfg -s ike:ikev2 listprop pkcs11_token/pin
pkcs11_token/pin     astring      PIN
```

- **When automatic keystore login is not configured, log in to the keystore manually.**

  Run this command each time the `in.ikev2d` daemon starts.

```
# pfbash
# ikeadm -v2 token login "Sun Metaslot"
Enter PIN for PKCS#11 token 'Sun Metaslot':      Type the PIN from Step 1
ikeadm: PKCS#11 operation successful
```

3. **(Optional) Verify that a PIN has been set in the keystore.**

```
# ikev2cert tokens
Flags: L=Login required  I=Initialized  X=User PIN expired  S=SO PIN expired
Slot ID     Slot Name                   Token Name                        Flags
-------     ---------                   ----------                        -----
1           Sun Crypto Softtoken        Sun Software PKCS#11 softtoken    LI
```

The `LI` in the `Flags` column indicates that the PIN is set.

4. **To manually log out of the `pkcs11_softtoken`, use the `ikeadm` command.**

```
# ikeadm -v2 token logout "Sun Metaslot"
ikeadm: PKCS#11 operation successful
```

You might log out to limit communication between two sites to a finite period of time. By logging out, the private key becomes unavailable, so new IKEv2 sessions cannot be initiated. The existing IKEv2 session continues unless you delete the session keys with the `ikeadm delete ikesa` command. Preshared key rules continue to work. See the `ikeadm(8)` man page.

## Configuring IKEv2 With Public Key Certificates

Public certificates can be a good choice for large deployments. For more information, see "IKE With Public Key Certificates" on page 132.

Public key certificates are stored in a softtoken keystore by the Cryptographic Framework.

For background information, see "How IKE Works" on page 131.

The following task map lists procedures for creating public key certificates for IKEv2.

**TABLE 10** Configuring IKEv2 With Public Key Certificates Task Map

| Task | Description | For Instructions |
|---|---|---|
| Create a keystore for certificates. | Initializes the PKCS #11 keystore where the certificates for IKEv2 are stored. | "Initializing the Keystore to Store Public Key Certificates for IKEv2" on page 149 |
| Configure IKEv2 with self-signed public key certificates. | Creates a public key certificate signed by you. Exports the certificate to peers and imports the peers' certificates. | "How to Configure IKEv2 With Self-Signed Public Key Certificates" on page 152 |
| Configure IKEv2 with a certificate from a CA. | Requires you to create a CSR and then import all returned certificates into the keystore. Then, verify and import the IKE peers' certificates. | "How to Configure IKEv2 With Certificates Signed by a CA" on page 158 |
| Configure how revoked certificates are handled. | Determines if CRLs are used and OCSP servers are polled, including how to handle network delays. | "How to Set a Certificate Validation Policy in IKEv2" on page 161 |

### ▼ How to Configure IKEv2 With Self-Signed Public Key Certificates

In this procedure, you create and sign a public key certificate. The private key and certificate are stored in the PKCS #11 softtoken keystore for IKEv2. You send the public key certificate to IKE peers, who in turn, send you their public certificate.

You perform this procedure on all IKE systems that use self-signed certificates.

**Before You Begin** To use the certificates, you must have completed "How to Create and Use a Keystore for IKEv2 Public Key Certificates" on page 150.

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be using a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Create a self-signed certificate in the keystore.**

   For a description of the arguments to the `ikev2cert gencert` command, review the `pktool gencert keystore=pkcs11` subcommand in the `pktool(1)` man page.

   For the format of the `subject` argument, see "Using Public Key Certificates in IKE" on page 132.

   **Note -** Give the certificate a label. The label identifies the certificate and its corresponding keys in the local keystore.

   a. **For example, the command on the `host2` system would appear similar to the following:**

```
# pfbash
 # ikev2cert gencert \
 label="IThost2" \
 subject="O=exampleco, OU=IT, C=US, CN=host2" \
 serial=0x87654321
    keytype=rsa
    keylen=2048
 Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

   b. **The command on the `host1` system would appear similar to the following:**

```
# ikev2cert gencert \
 label=IThost1 \
 subject="O=exampleco, OU=IT, C=US, CN=host1" \
 serial=0x86428642
    keytype=rsa
    keylen=2048
 Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

2. **(Optional) List the keys and certificate.**

```
host1 # /usr/sbin/ikev2cert list objtype=both
 Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
 No.     Key Type        Key Len.        Key Label
 ----------------------------------------------------
 Asymmetric private keys:
 1)      RSA                             IThost1
 Asymmetric public keys:
 1)      RSA                             IThost1
 Certificates:
 1) X.509 certificate
  Label: IThost1
  Subject: C=US, O=exampleco, OU=IT, CN=host1
  Issuer: C=US, O=exampleco, OU=IT, CN=host1
  Not Before: April 10 21:49:00 2014 GMT
  Not After: April 10 21:49:00 2015 GMT
  Serial: 0x86426420
  Signature Algorithm: sha1WithRSAEncryption
  X509v3 Subject Key Identifier:
   34:7a:3b:36:c7:7d:4f:60:ed:ec:4a:96:33:67:f2:ac:87:ce:35:cc
  SHA1 Certificate Fingerprint:
   68:07:48:65:a2:4a:bf:18:f5:5b:95:a5:01:42:c0:26:e3:3b:a5:30
```

**Tip -** The default hash algorithm is `SHA1`. To create a certificate with a stronger signature algorithm, use the `keytype` option and a different hash algorithm, such as `keytype=rsa hash=sha384`. For the options, see the `pktool`(1) man page.

3. **Deliver the certificate to the other system.**

   a. **On each system, export only the certificate to a file.**

   The `outformat=pem` option ensures that the public certificate is placed in the file in a format that is suitable for direct import. The label identifies the certificate in the keystore.

```
# cd /tmp
 # ikev2cert export objtype=cert outformat=pem outfile=filename label=label
 Enter PIN for Sun Software PKCS#11 softtoken:xxxxxxxx
```

   b. **Send the certificate to the other system by email, `sftp`, or `ssh`.**

   For example, if you administer both systems, use the `sftp` command to bring over the certificate from the other system.

```
host1 # sftp jdoe@host2:/tmp/IThost2.pem /tmp/IThost2.pem.cert
 host2 # sftp jdoe@host1:tmp/IThost1.pem /tmp/IThost1.pem.cert
```

You are prompted for your password. In this example, `jdoe` must provide a password.

4. **Verify that the certificates are identical.**

   You want to ensure that you have received the proper certificate *before* you load it into your keystore.

   a. **Create a digest of the exported file on each system.**

      For example, the `host2` administrator emails the digest of the file that contains `host2`'s certificate to the other administrator. The `host1` administrator emails the digest of the `host1` certificate file.

      ```
      host2 # digest -a sha1 /tmp/IThost2.pem
       c6dbef4136c0141ae62110246f288e5546a59d86

       host1 # digest -a sha1 IThost1.pem
       6b288a6a6129d53a45057065bd02b35d7d299b3a
      ```

   b. **On the other system, run the `digest` command on the file that contains the certificate from the first system.**

      ```
      host1 # digest -a sha1 /tmp/IThost2.pem.cert
       c6dbef4136c0141ae62110246f288e5546a59d86

       host2 # digest -a sha1 /tmp/IThost1.pem.cert
       6b288a6a6129d53a45057065bd02b35d7d299b3a
      ```

      The digests must match. If they do not match, do not import the file to the keystore. For another way of verifying certificate validity, see Example 31, “Verifying a Public Key Certificate by Its Fingerprint,” on page 158.

5. **After verification, import the other system's certificate to your keystore.**

   When you import the certificate into your keystore, you must assign a label to it that uniquely identifies the certificate on your system. The label links the public key with its public key certificate.

   ```
   host1 # ikev2cert import label=IThost2 infile=/tmp/IThost2.pem.cert

   host2 # ikev2cert import label=IThost1 infile=/tmp/IThost1.pem.cert
   ```

6. **(Optional) List the objects in your keystore.**

   Compare the listing with the list in Step 2. For example, in the `host1` keystore, the `host2` certificate is added.

```
host1 # /usr/sbin/ikev2cert list objtype=both
 Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
 No.     Key Type        Key Len.        Key Label
 ----------------------------------------------------
 Asymmetric private keys:
 1)      RSA                             IThost1
 Asymmetric public keys:
 1)      RSA                             IThost1
 Certificates:
 1) X.509 certificate
  Label: IThost1
  Subject: C=US, O=exampleco, OU=IT, CN=host1
  Issuer: C=US, O=exampleco, OU=IT, CN=host1
  Not Before: April 10 21:49:00 2014 GMT
  Not After: April 10 21:49:00 2015 GMT
  Serial: 0x86426420
  Signature Algorithm: sha1WithRSAEncryption
  X509v3 Subject Key Identifier:
   34:7a:3b:36:c7:7d:4f:60:ed:ec:4a:96:33:67:f2:ac:87:ce:35:cc
  SHA1 Certificate Fingerprint:
   68:07:48:65:a2:4a:bf:18:f5:5b:95:a5:01:42:c0:26:e3:3b:a5:30

 2) X.509 certificate
  Label: IThost2
  Subject: C=US, O=exampleco, OU=IT, CN=host2
  Issuer: C=US, O=exampleco, OU=IT, CN=host2
  Not Before: April 10 21:40:00 2014 GMT
  Not After: April 10 21:40:00 2015 GMT
  Serial: 0x87654321
  Signature Algorithm: sha1WithRSAEncryption
  X509v3 Subject Key Identifier:
   ae:d9:c8:a4:19:68:fe:2d:6c:c2:9a:b6:06:55:b5:b5:d9:d9:45:c6
  SHA1 Certificate Fingerprint:
   83:26:44:29:b4:1f:af:4a:69:0d:87:c2:dc:f4:a5:1b:4f:0d:36:3b
```

7. **On each system, use the certificates in an IKEv2 rule.**

   Use the `pfedit` command to edit the `/etc/inet/ike/ikev2.config` file.

   a. **For example, on the `host2` system, the rule in the `ikev2.config` file would appear similar to the following:**

```
##  ... Global transform that applies to any rule without a declared transform
 ikesa_xform { dh_group 21 auth_alg sha512 encr_alg aes }
 ##  ... Any self-signed
 ## end-entity certificates must be present in the keystore or
 ## they will not be trusted.
```

```
{
 label "host2-host1"
 auth_method cert
 local_id DN = "O=exampleco, OU=IT, C=US, CN=host2"
 remote_id DN = "O=exampleco, OU=IT, C=US, CN=host1"
}
...
```

b. **On the `host1` system, use the DN of the `host1` certificate for the value of `local_id` in the `ikev2.config` file.**

   For the remote parameter, use the `host2` certificate's DN as the value. Ensure that the value for the `label` keyword is unique on the local system.

```
…
 ikesa_xform { dh_group 21 auth_alg sha512 encr_alg aes }
 …
 {
  label "host1-host2"
  auth_method cert
  local_id DN = "O=exampleco, OU=IT, C=US, CN=host1"
  remote_id DN = "O=exampleco, OU=IT, C=US, CN=host2"
 }
 ...
```

8. **(Optional) On each system, check the validity of the `ikev2.config` files.**

```
# /usr/lib/inet/inikev2.d -c
```

   Fix any typographical errors or inaccuracies before continuing.

9. **On each system, check the state of the IKEv2 service instance.**

```
# svcs ikev2
 STATE          STIME    FMRI
 disabled       Sep_07   svc:/network/ipsec/ike:ikev2
```

10. **On each system, enable the IKEv2 service instance.**

```
host2 # svcadm enable ipsec/ike:ikev2

 host1 # svcadm enable ipsec/ike:ikev2
```

**Example 30** Creating a Self-Signed Certificate With a Limited Lifetime

In this example, the administrator specifies that the certificate is valid for two years.

```
# ikev2cert gencert \
> label=DBAuditV \
> serial=0x12893467235412 \
> subject="O=exampleco, OU=DB, C=US, CN=AuditVault" \
> altname=EMAIL=auditV@example.com \
> keytype=ec curve=secp521r1 hash=sha512 \
   > lifetime=2-year
```

**Example 31** Verifying a Public Key Certificate by Its Fingerprint

In this example, the administrator uses the certificate fingerprint to verify the certificate. The disadvantage of this method is that the administrator must import the peer's certificate into the keystore before viewing the fingerprint.

The administrator imports the certificate, lists it with the `ikev2cert list objtype=cert` command, then copies the certificate fingerprint from the output and sends it to the other system administrator.

```
SHA1 Certificate Fingerprint:
             83:26:44:29:b4:1f:af:4a:69:0d:87:c2:dc:f4:a5:1b:4f:0d:36:3b
```

If the verification fails, the administrator who imported the certificate must remove it and its public key from the keystore.

```
# ikev2cert delete label=label-name
       Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
       1 public key(s) found, do you want to delete them (y/N) ? y
       1 certificate(s) found, do you want to delete them (y/N) ? y
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Configure IKEv2 With Certificates Signed by a CA

Organizations that protect a large number of communicating systems typically use public certificates from a certificate authority (CA). For background information, see "IKE With Public Key Certificates" on page 132.

You perform this procedure on all IKE systems that use certificates from a CA.

**Before You Begin** To use the certificates, you must have completed "How to Create and Use a Keystore for IKEv2 Public Key Certificates" on page 150.

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Change to a writable directory.**

   The following error message can indicate that the CSR file cannot be written to disk:

   ```
   Warning: error accessing "CSR-file"
   ```

   For example, use the `/tmp` directory.

   ```
   # cd /tmp
   ```

2. **Create a certificate signing request.**

   You use the `ikev2cert gencsr` command to create a certificate signing request (CSR). For a description of the arguments to the command, review the `pktool gencsr keystore=pkcs11` subcommand in the `pktool(1)` man page.

   For example, the following command creates a file that contains the CSR on the `host2` system:

   ```
   # pfbash
   # /usr/sbin/ikev2cert gencsr \
   keytype=rsa
   keylen=2048
   label=Example2m \
   outcsr=/tmp/Example2mcsr1 \
   subject="C=US, O=Example2Co\, Inc., OU=US-Example1m, CN=Example1m"
   Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
   ```

3. **(Optional) Copy the contents of the CSR for pasting into the CA's web form.**

   ```
   # cat /tmp/Example2mcsr1
   -----BEGIN CERTIFICATE REQUEST-----
   MIICkDCCAXoCAQAwTzELMAkGA1UEBhMCVVMxGzAZBgNVBAoTElBhcnR5Q29tcGFu
   eSwgSW5jLjESMBAGA1UECxMJVVMtUGFydHltMQ8wDQYDVQQDEwZQYXJ0eW0wggEi
   MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCMbINmgZ4XWUv2q1fshZUN/SLb
   WNLXZxdKwt5e71o0owjyby69eL7HE0QBUij73nTkXE3n4gxojBZE+hvJ6GOCbREA
   jgSquP2US7Bn9XEcXRrsOc7MCFPrsA+hVIcNHpKNseUOU/rg+wzoo5hA1ixtWuXH
   bYDeEWQi5tlZgDZoCWGrdHEjwVyHfvz+a0WBjyZBYOueBhXaa68QqSOSnRVDX56Q
   3p4H/AR4h0dcSja72XmMKPU5p3RVb8n/hrfKjiDjiGYXD4D+WZxQ65xxCcnALvVH
   ```

```
nZHUlAtP7QHX4RXlQVNNwEsY6C95RX9297rNWLsYvp/86xWrQkTlNqVAeUKhAgMB
AAEwCwYJKoZIhvcNAQEFA4IBAQB3R6rmZdqcgN8Tomyjp2CFTdyAWixkIATXpLM1
GL5ghrnDvadD61M+vS1yhFlIcSNM8fLRrCHIKtAmB8ITnggJ//rzbHq3jdla/iQt
kgGoTXfz8j6B57Ud6l+MBLiBSBy0QK4GIg8Ojlb9Kk5HsZ48mIoI/Qb7FFW4p9dB
JEUn0eYhkaGtwJ21YNNvKgOeOcnSZy+xP9Wa9WpfdsBO4TicLDw0Yq7koNnfL0IB
Fj2bt/wI7iZ1DcpwphsiwnW9K9YynAJZzHd1ULVpn5Kd7vSRz9youLLzSb+9ilgO
E43DW0hRk6P/Uq0N4e1Zca4otezNxyEqlPZI7pJ5uOo0sbiw
-----END CERTIFICATE REQUEST-----
```

4. **Submit the CSR to a certificate authority (CA).**

   The CA can tell you how to submit the CSR. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your CSR into the form.

   ---

   **Tip -** Some web forms have an Advanced button where you can paste your certificate. The CSR is generated in PKCS#10 format. Therefore, find the portion of the web form that mentions PKCS#10.

   ---

5. **Import each certificate that you receive from the CA into your keystore.**

   The `ikev2cert import` imports the certificate into the keystore.

   a. **Import the public key and certificate that you received from the CA.**

   ```
   # ikev2cert import objtype=cert label=Example1m1 infile=/tmp/Example1m1Cert
   ```

   ---

   **Tip -** For administrative convenience, assign the same label to the imported certificate as the label of the original CSR.

   ---

   b. **Import the root certificate from the CA.**

   ```
   # ikev2cert import objtype=cert infile=/tmp/Example1m1CAcert
   ```

   c. **Import any intermediate CA certificates into the keystore.**

   ---

   **Tip -** For administrative convenience, assign the same label to the imported intermediate certificates as the label of the original CSR.

   ---

   If the CA has sent separate files for each intermediate certificate, then import them as you imported the preceding certificates. However, if the CA delivers its certificate chain as a PKCS#7 file, you must extract the individual certificates from the file, then import each certificate as you imported the preceding certificates:

**Note -** You must assume the `root` role to run the `openssl` command. See the `openssl`(7) man page.

```
# openssl pkcs7 -in pkcs7-file -print_certs
# ikev2cert import objtype=cert label=Example1m1 infile=individual-cert
```

6. **Set the certificate validation policy.**

   If the certificate contains sections for CRLs or OCSP, you must configure the certificate validation policy according to your site requirements. For instructions, see "How to Set a Certificate Validation Policy in IKEv2" on page 161.

7. **After you complete the procedure on all IKE systems which use your certificate, enable the `ikev2` service on all systems.**

   The peer systems need the root certificate and a configured `ikev2.config` file.

**Example 32** Using a FIPS 140-2 Approved ECC Algorithm in an IKEv2 CSR

The following excerpt from an `ikev2cert` command generates a certificate request with one of the FIPS 140-2 approved ECC algorithms, using curve `secp521r1` and hash `sha512`:

```
# ikev2cert gencsr label=FIPSokcsr \
    subject="C=Country, O=Company\, Inc., OU=CompanyServer, CN=Server" \
    keytype=ec curve=secp521r1 hash=sha512 \
    outcsr=/tmp/FIPSokcsr
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Set a Certificate Validation Policy in IKEv2

You can configure several aspects of how certificates are handled for your IKEv2 system.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Review the default certificate validation policy.**

   Certificate policy is set at installation in the `/etc/inet/ike/kmf-policy.xml` file. The file is owned by `ikeuser` and is modified by using the `kmfcfg` command. The default certificate validation policy is to download CRLs to the `/var/user/ikeuser/crls` directory. The use of OCSP is also enabled by default. If your site requires a proxy to reach the Internet, you must configure the proxy. See "How to Handle Revoked Certificates in IKEv2" on page 163.

```
# pfbash
# kmfcfg list dbfile=/etc/inet/ike/kmf-policy.xml policy=default
Policy Name: default
Ignore Certificate Validity Dates: false     Unknown purposes or applications for the certificate
Ignore Unknown EKUs: false
Ignore Trust Anchor in Certificate Validation: false
Trust Intermediate CAs as trust anchors: false
Maximum Certificate Path Length: 32
Certificate Validity Period Adjusted Time leeway: [not set]
Trust Anchor Certificate: Search by Issuer
Key Usage Bits: 0     Identifies critical parts of certificate
Extended Key Usage Values: [not set]     Purposes or applications for the certificate
HTTP Proxy (Global Scope): [not set]
Validation Policy Information:
    Maximum Certificate Revocation Responder Timeout: 10
    Ignore Certificate Revocation Responder Timeout: true
    OCSP:
        Responder URI: [not set]
        OCSP specific proxy override: [not set]
        Use ResponderURI from Certificate: true
        Response lifetime: [not set]
        Ignore Response signature: false
        Responder Certificate: [not set]
    CRL:
        Base filename: [not set]
        Directory: /var/user/ikeuser/crls
        Download and cache CRL: true
        CRL specific proxy override: [not set]
        Ignore CRL signature: false
        Ignore CRL validity date: false
IPsec policy bypass on outgoing connections: true
Certificate to name mapper name: [not set]
Certificate to name mapper pathname: [not set]
Certificate to name mapper directory: [not set]
Certificate to name mapper options: [not set]
```

2. **Review the certificate for features that indicate the validation options to modify.**

   For example, a certificate that includes a CRL or OCSP URI can use a validation policy that specifies the URI to use to check certificate revocation status. You might also configure timeouts.

3. **Review the `kmfcfg`(1) man page for configurable options.**

4. **Configure the certificate validation policy.**

   For a sample policy, see "How to Handle Revoked Certificates in IKEv2" on page 163.

## ▼ How to Handle Revoked Certificates in IKEv2

Revoked certificates are certificates that are compromised for some reason. A revoked certificate that is in use is a security risk. You have options when verifying certificate revocation. You can use a static list or you can verify revocations dynamically over the HTTP protocol.

**Before You Begin** You have received and installed certificates from a CA.

You are familiar with the CRL and OCSP methods of checking for revoked certificates. For information and pointers, see "IKE With Public Key Certificates" on page 132.

You must become an administrator who is assigned the Network IPsec Management rights profile, and use a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Locate the CRL and OCSP sections in the certificate that you received from the CA.**

   You can identify the certificate from the label of your CSR.

```
$ pfbash
$ ikev2cert list objtype=cert | grep Label:
Enter PIN for Sun Software PKCS#11 softtoken:
        Label: Example1m1
```

   For example, the following truncated output highlights the CRL and OCSP URIs in a certificate.

```
$ ikev2cert list objtype=cert label=Example1m1
X509v3 extensions:
    ...
    X509v3 CRL Distribution Points:
         Full Name:
       URI:http://onsitecrl.PKI.example.com/OCCIPsec/LatestCRL.crl
    X509v3 Authority Key Identifier:
         ...
    Authority Information Access:
        OCSP - URI:http://ocsp.PKI.example.com/revokes/
    X509v3 Certificate Policies:
         Policy: 2.16.840.1.113733.1.7.23.2
```

Under the `CRL Distribution Points` entry, the URI value indicates that this organization's CRL is available in a file on the web. The OCSP entry indicates that the status of individual certificates can be determined dynamically from a server.

2. **Enable the use of CRLs or an OCSP server by specifying a proxy.**

```
# kmfcfg modify \
dbfile=/etc/inet/ike/kmf-policy.xml \
policy=default \
http-proxy=www-proxy.ja.example.com:80
```

At sites where a proxy is optional, you do not need to specify one.

3. **Verify that the certificate validation policy is updated.**

For example, verify that the OCSP was updated.

```
$ kmfcfg list \
dbfile=/etc/inet/ike/kmf-policy.xml \
policy=default
...
    OCSP:
        Responder URI: [not set]
        Proxy: www-proxy.ja.example.com:80
        Use ResponderURI from Certificate: true
        Response lifetime: [not set]
        Ignore Response signature: false
        Responder Certificate: [not set]
```

4. **Restart the IKEv2 service.**

```
$ pfexec svcadm restart ikev2
```

5. **(Optional) Stop using CRLs or OCSP.**

- **To stop using CRLs, type:**

```
$ pfexec kmfcfg modify \
dbfile=/etc/inet/ike/kmf-policy.xml policy=default \
crl-none=true
```

The `crl-none=true` argument forces the system to use downloaded CRLs from the local cache.

- **To stop using OCSP, type:**

```
$ pfexec kmfcfg modify \
dbfile=/etc/inet/ike/kmf-policy.xml policy=default \
ocsp-none=true
```

**Example 33** Changing the Time That a System Waits For IKEv2 Certificate Verification

In this example, the administrator limits the wait to twenty seconds for a certificate to be verified.

```
# kmfcfg modify dbfile=/etc/inet/ike/kmf-policy.xml policy=default \
    cert-revoke-responder-timeout=20
```

By default, when a response times out, the authentication of the peer succeeds. Here, the administrator configures a policy where the connection is refused when authentication fails. In this configuration, certificate validation fails when an OCSP or CRL server becomes unresponsive.

```
# kmfcfg modify dbfile=/etc/inet/ike/kmf-policy.xml policy=default \
    ignore-cert-revoke-responder-timeout=false
```

To activate the policy, the administrator restarts the IKEv2 service.

```
# svcadm restart ikev2
```

♦ ♦ ♦ CHAPTER 10

# Configuring IKEv1

This chapter describes how to configure the Internet Key Exchange version 1 (IKEv1) for your systems. After IKEv1 is configured, it automatically generates keying material for IPsec on your network. This chapter contains the following information:

- "Configuring IKEv1 With Preshared Keys" on page 168
- "Configuring IKEv1 With Public Key Certificates" on page 173
- "Configuring IKEv1 for Mobile Systems" on page 187

---

**Note -** If you plan to implement IKEv2 only, proceed to Chapter 9, "Configuring IKEv2".

---

For overview information about IKE, see Chapter 8, "About Internet Key Exchange". For reference information about IKE, see Chapter 12, "IPsec and Key Management Reference". For more procedures, see the Examples sections of the `ikeadm`(8), `ikecert`(8), and `ike.config`(5) man pages.

## Configuring IKEv1

You can use preshared keys, self-signed certificates, and certificates from a certificate authority (CA) to authenticate IKE. A rule in the `ike/config` file links the particular IKEv1 authentication method with the IKEv1 peer. Therefore, you can use one or all IKE authentication methods on a system.

After configuring IKEv1, complete the IPsec task in Chapter 7, "Configuring IPsec" that uses the IKEv1 configuration.

## Configuring IKEv1 With Preshared Keys

If you are configuring peer systems or subnets to use IKEv1 and you are the administrator of these subnets, using preshared keys can be a good choice. Preshared keys might also be used when testing. For more information, see "IKE With Preshared Key Authentication" on page 131.

### ▼ How to Configure IKEv1 With Preshared Keys

The IKE implementation offers algorithms whose keys vary in length. The key length that you choose is determined by site security. In general, longer keys provide more security than shorter keys.

In this procedure, you generate keys in ASCII format.

These procedures use the system names `host1` and `host2`. Substitute the names of your systems for the names `host1` and `host2`.

---

**Note -** To use IPsec with labels on a Trusted Extensions system, see the extension of this procedure in "How to Apply IPsec Protections in a Multilevel Trusted Extensions Network" in *Trusted Extensions Configuration and Administration*.

---

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **On each system, create an `/etc/inet/ike/config` file.**

   You can use the `/etc/inet/ike/config.sample` as a template.

2. **Enter rules and global parameters in the `ike/config` file on each system.**

   The rules and global parameters in this file should permit the IPsec policy in the system's `ipsecinit.conf` file to succeed. The following IKEv1 configuration examples work with the `ipsecinit.conf` examples in "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

a. **For example, modify the `/etc/inet/ike/config` file on the `host1` system:**

```
### ike/config file on host1, 192.0.2.16

## Global parameters
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 14 auth_alg sha encr_alg 3des }
p2_pfs 14
#
## The rule to communicate with host2
#  Label must be unique
{ label "host1-host2"
  local_addr 192.0.2.16
  remote_addr 192.0.2.213
  p1_xform
   { auth_method preshared oakley_group 14 auth_alg sha256 encr_alg aes }
  p2_pfs 14
}
```

b. **Modify the `/etc/inet/ike/config` file on the `host2` system:**

```
### ike/config file on host2, 192.0.2.213
## Global Parameters
#
p1_xform
  { auth_method preshared oakley_group 14 auth_alg sha encr_alg 3des }
p2_pfs 14

## The rule to communicate with host1
#  Label must be unique
{ label "host2-host1"
  local_addr 192.0.2.213
  remote_addr 192.0.2.16
p1_xform
 { auth_method preshared oakley_group 14 auth_alg sha256 encr_alg aes }
p2_pfs 14
}
```

3. **On each system, verify the syntax of the file.**

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

4. **Create a preshared key for IKEv1 to use.**

An AES key of at least 256 bits is a good choice. For a full description of how to create a key, see "How to Generate a Symmetric Key by Using the pktool Command" in *Managing Encryption and Certificates in Oracle Solaris 11.4*. For examples of key generation, see Example 28, "Generating a Preshared Key for IKEv2," on page 145 and Example 29, "Using Different Local and Remote IKEv2 Preshared Keys," on page 145.

5. **Put the preshared key in the `/etc/inet/secret/ike.preshared` file on each system.**

   a. **For example, on the `host1` system, the `ike.preshared` file would appear similar to the following:**

```
## ike.preshared on host1, 192.0.2.16
#...
{ localidtype IP
 localid 192.0.2.16
 remoteidtype IP
 remoteid 192.0.2.213
 # The two subnet's shared hex key
   key  "1011e1f2d1fd..."
}
```

   b. **On the `host2` system, the `ike.preshared` file would appear similar to the following:**

```
## ike.preshared on host2, 192.0.2.213
#...
{ localidtype IP
 localid 192.0.2.213
 remoteidtype IP
 remoteid 192.0.2.16
 # The two subnet's shared hex key
   key  "1011e1f2d1fd..."
 }
```

6. **Enable the IKEv1 service.**

```
# svcadm enable ipsec/ike:default
```

**Example 34** Refreshing an IKEv1 Preshared Key

When IKEv1 administrators want to refresh the preshared key, they edit the files on the peer systems and restart the `in.iked` daemon.

First, on every system in the two subnets that uses the preshared key, the administrator changes the preshared key entry.

```
# pfedit -s /etc/inet/secret/ike.preshared
...
{ localidtype IP
 localid 192.0.2.0/27
 remoteidtype IP
 remoteid 192.0.2.32/27
 # The two subnet's shared hex key
      key "f8be7576851573..."
 }
```

Then, the administrator restarts the IKEv1 service on every system.

For information about the options to the `pfedit` command, see the `pfedit`(8) man page.

```
# svcadm enable ipsec/ike:default
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Update IKEv1 for a New Peer System

If you add IPsec policy entries to a working configuration between the same peers, you need to refresh the IPsec policy service. You do not need to reconfigure or restart IKEv1.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKEv1 configuration.

**Before You Begin** You have updated the `ipsecinit.conf` file and refreshed IPsec policy for the peer systems.

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Create a rule for IKEv1 to manage the keys for the new system that is using IPsec.**

   a. **For example, on the `host1` system, add the following rule to the `/etc/inet/ike/config` file:**

```
### ike/config file on host1, 192.0.2.16
...
## The rule to communicate with host3

{label "host1-to-host3"
 local_addr 192.0.2.16
 remote_addr 192.0.2.7
 p1_xform
 {auth_method preshared oakley_group 14 auth_alg sha256 encr_alg aes}
 p2_pfs 14
 }
```

b. **On the `host3` system, add the following rule:**

```
### ike/config file on host3, 192.0.2.7

## The rule to communicate with host1

{label "host3-to-host1"
 local_addr 192.0.2.7
 remote_addr 192.0.2.16
 p1_xform
 {auth_method preshared oakley_group 14 auth_alg sha256 encr_alg aes}
 p2_pfs 14
}
```

2. **Create an IKEv1 preshared key for the peer systems.**

a. **On the `host1` system, add the following information to the `/etc/inet/secret/ike.preshared` file:**

```
## ike.preshared on host1 for the host3 interface
##
{ localidtype IP
  localid 192.0.2.16
  remoteidtype IP
  remoteid 192.0.2.7
  # host1 and host3's shared hex key
    key "2b823670b5aa1a..."
}
```

b. **On the `host3` system, add the following information to the `ike.preshared` file:**

```
## ike.preshared on host3 for the host1 interface
##
{ localidtype IP
```

```
        localid 192.0.2.7
        remoteidtype IP
        remoteid 192.0.2.16
        # host3 and host1's shared hex key
          key "2b823670b5aa1a..."
      }
```

3. **On each system, refresh the `ike` service.**

   ```
   # svcadm refresh ike:default
   ```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## Configuring IKEv1 With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Public certificates from a certificate authority (CA) typically require negotiation with an outside organization. The certificates very easily scale to protect a large number of communicating systems.

All certificates have a unique name in the form of an X.509 distinguished name (DN). Additionally, a certificate might have one or more subject alternative names, such as an email address, a DNS name, an IP address, and so on. You can identify the certificate in the IKEv1 configuration by its full DN or by one of its subject alternative names. The format of these alternative names is *tag=value*, where the format of the value corresponds to its tag type. For example, the format of the `email` tag is *name@domain.suffix*.

The following task map lists procedures for creating public key certificates for IKEv1.

**TABLE 11** Configuring IKEv1 With Public Key Certificates Task Map

| Task | Description | For Instructions |
|---|---|---|
| Configure IKEv1 with self-signed public key certificates. | Creates and places keys and two certificates on each system:<br>■ A self-signed certificate and its keys<br>■ The public key certificate from the peer system | "How to Configure IKEv1 With Self-Signed Public Key Certificates" on page 174 |
| Configure IKEv1 with a certificate authority. | Creates a certificate signing request, and then places certificates from the CA on each system. See "Using Public Key Certificates in IKE" on page 132. | "How to Configure IKEv1 With Certificates Signed by a CA" on page 179 |

| Task | Description | For Instructions |
|---|---|---|
| Update the certificate revocation list (CRL) from the CA. | Accesses the CRL from a central distribution point. | "How to Handle Revoked Certificates in IKEv1" on page 185 |

**Note -** To label packets and IKE negotiations on a Trusted Extensions system, follow the procedures in "Configuring Labeled IPsec" in *Trusted Extensions Configuration and Administration*.

Public key certificates are managed in the global zone on Trusted Extensions systems. Trusted Extensions does not change how certificates are managed and stored.

## ▼ How to Configure IKEv1 With Self-Signed Public Key Certificates

In this procedure, you create a public/private key and a certificate, called a certificate pair. The private key is stored on disk in the local certificate database and can be referenced by using the `ikecert certlocal` command. The public key and certificate is stored in the public certificate database. It can be referenced by using the `ikecert certdb` command. You exchange the public certificate with a peer system. The two certificates are used to authenticate the IKEv1 transmissions.

Self-signed certificates require less overhead than public certificates from a CA, but do not scale very easily. Unlike certificates that are issued by a CA, self-signed certificates must be verified by the two administrators who exchanged the certificates.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **On each IKEv1 system, create a self-signed certificate in the `ike.privatekeys` database.**

   For arguments to the `ikecert certlocal` command, see the `ikecert`(8) man page.

   a. **For example, the command on the `host2` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 2048 -t rsa-sha512 \
-D "O=exampleco, OU=IT, C=US, CN=host2" \
-A IP=192.0.2.213
Creating private key.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
a...+
zBGi4QkNdI3f
-----END X509 CERTIFICATE-----
```

where

| | |
|---|---|
| `-ks` | Creates a self-signed certificate. |
| `-m` *keysize* | Specifies the size of the key. |
| `-t` *keytype* | Specifies the type of algorithm to use. |
| `-D` *dname* | Specifies the X.509 distinguished name (DN) for the certificate subject. For an example, see "Using Public Key Certificates in IKE" on page 132. |
| `-A` *altname* | Specifies the alternate name or nickname for the certificate. The *altname* is in the form of *tag=value*. Valid tags are `IP`, `DNS`, `email`, and `DN`. |

---

**Note -** The values of the `-D` and `-A` options are names that identify the certificate only, not any system, such as `192.0.2.213`. In fact, because these values are certificate nicknames, you must verify out of band that the correct certificate is installed on the peer systems.

---

**b. The command on the `host1` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 2048 -t rsa-sha512 \
-D "O=exampleco, OU=IT, C=US, CN=host1" \
-A IP=192.0.2.16
Creating private key.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEBl5JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
...
y85m6LHJYtC6
-----END X509 CERTIFICATE-----
```

2. **Save the certificate and send it to the remote system.**

   The output is an encoded version of the public portion of the certificate. You can safely paste this certificate into an email message. The receiving party must verify out of band that they installed the correct certificate, as shown in Step 4.

   a. **For example, you would send the public portion of the `host2` certificate to the `host1` administrator.**

   ```
   To: admin@host1.ja.example.com
   From: admin@host2.us.example.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
   a...+
   zBGi4QkNdI3f
   -----END X509 CERTIFICATE------
   ```

   b. **The `host1` administrator would send you the public portion of the `host1` certificate.**

   ```
   To: admin@host2.us.example.com
   From: admin@example.ja.example.com
   Message: ----BEGIN X509 CERTIFICATE-----
   MIIC1TCCAb2gAwIBAgIEBl5JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
   ...
   y85m6LHJYtC6
   -----END X509 CERTIFICATE-----
   ```

3. **On each system, add the certificate that you received to the public key database.**

   a. **Save the administrator's email to a file that is read by `root`.**

   b. **Redirect the file to the `ikecert` command.**

   ```
   # ikecert certdb -a < /tmp/certificate.eml
   ```

   The command imports the text between the `BEGIN` and `END` tags.

4. **Verify with the other administrator that the certificate is from that administrator.**

   For example, you can telephone the other administrator to verify that the hash of their public certificate, which you have, matches the hash of their private certificate, which only they have.

   a. **List the stored certificate on `host2`.**

   In the following example, Note 1 indicates the distinguished name (DN) of the certificate in slot 0. The private certificate in slot 0 has the same hash (see Note 3), so these certificates are the same certificate pair. For the public certificates to work, you must have a matching pair. The `certdb` subcommand lists the public portion, while the `certlocal` subcommand lists the private portion.

```
host2 # ikecert certdb -l

Certificate Slot Name: 0   Key Type: rsa
 (Private key in certlocal slot 0)
 Subject Name: <O=exampleco, OU=IT, C=US, CN=host2>    Note 1
 Key Size: 2048
 Public key hash: 80829EC52FC5BA910F4764076C20FDCF

Certificate Slot Name: 1   Key Type: rsa
 (Private key in certlocal slot 1)
 Subject Name: <O=exampleco, OU=IT, C=US, CN=Ada>
 Key Size: 2048
 Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388
host2 # ikecert certlocal -l
Local ID Slot Name: 0   Key Type: rsa
 Key Size: 2048
 Public key hash: 80829EC52FC5BA910F4764076C20FDCF    Note 3

Local ID Slot Name: 1   Key Type: rsa-sha512
        Key Size: 2048
        Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388

Local ID Slot Name: 2   Key Type: rsa
 Key Size: 2048
 Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

This check has verified that the `host2` system has a valid certificate pair.

b. **Verify that the `host1` system has `host2`'s public certificate.**

You can read the public key hash over the telephone.

Compare the hashes from Note 3 on `host2` in the preceding step with Note 4 on `host1`.

```
host1 # ikecert certdb -l

Certificate Slot Name: 0   Key Type: rsa
        (Private key in certlocal slot 0)
        Subject Name: <O=exampleco, OU=IT, C=US, CN=Ada>
        Key Size: 2048
        Public key hash: 2239A6A127F88EE0CB40F7C24A65B818

Certificate Slot Name: 1   Key Type: rsa
        (Private key in certlocal slot 1)
        Subject Name: <O=exampleco, OU=IT, C=US, CN=host1>
        Key Size: 2048
        Public key hash: FEA65C5387BBF3B2C8F16C019FEBC388

Certificate Slot Name: 2   Key Type: rsa
        (Private key in certlocal slot 2)
        Subject Name: <O=exampleco, OU=IT, C=US, CN=host2>
        Key Size: 2048
        Public key hash: 80829EC52FC5BA910F4764076C20FDCF    Note 4
```

The public key hash and subject name of the last certificate stored in `host1`'s public certificate database match the private certificate for `host2` from the preceding step.

5. **On each system, trust both certificates.**

   Edit the `/etc/inet/ike/config` file to recognize the certificates.

   The administrator of the remote system provides the values for the `cert_trust`, `remote_addr`, and `remote_id` parameters.

   a. **For example, on the `host2` system, the `ike/config` file would appear similar to the following:**

```
# Explicitly trust the self-signed certs
# that we verified out of band. The local certificate
# is implicitly trusted because we have access to the private key.

cert_trust "O=exampleco, OU=IT, C=US, CN=host1"
# We could also use the Alternate name of the certificate,
# if it was created with one.  In this example, the Alternate Name
# is in the format of an IP address:
# cert_trust "192.0.2.16"

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 14 auth_alg sha256 encr_alg 3des }
p2_pfs 14

{
 label "US-host2 to JA-host1;"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=host2"
 remote_id "O=exampleco, OU=IT, C=US, CN=host1"
 local_addr  192.0.2.213
# We could explicitly enter the peer's IP address here, but we don't need
# to do this with certificates, so use a wildcard address. The wildcard
# allows the remote device to be mobile or behind a NAT box
```

```
 remote_addr 0.0.0.0/0


 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

b. **On the `host1` system, add `host1` values for local parameters in the `ike/config` file.**

For the remote parameters, use `host2` values. Ensure that the value for the `label` keyword is unique on the local system.

```
...
{
 label "JA-host1 to US-host2"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=host1"
 remote_id "O=exampleco, OU=IT, C=US, CN=host2"

 local_addr  192.0.2.16
 remote_addr 0.0.0.0/0

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

6. **On the peer systems, enable IKEv1.**

```
host2 # svcadm enable ipsec/ike:default
host1 # svcadm enable ipsec/ike
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Configure IKEv1 With Certificates Signed by a CA

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Use the `ikecert certlocal -kc` command to create a certificate signing request (CSR).**

   For a description of the arguments to the command, see Step 1 in "How to Configure IKEv1 With Self-Signed Public Key Certificates" on page 174.

   ```
   # ikecert certlocal -kc -m keysize -t keytype \
   -D dname -A altname
   ```

   a. **For example, the following command creates a CSR on the `host2` system:**

   ```
   # ikecert certlocal -kc -m 2048 -t rsa-sha384 \
   > -D "C=US, O=Example2Co\, Inc., OU=US-Example2m, CN=Example2m" \
   > -A "DN=C=US, O=Example2Co\, Inc., OU=US-Example2m"
   Creating software private keys.
     Writing private key to file /etc/inet/secret/ike.privatekeys/2.
   Enabling external key providers - done.
   Certificate Request:
     Proceeding with the signing operation.
     Certificate request generated successfully (.../publickeys/0)
   Finished successfully.
   -----BEGIN CERTIFICATE REQUEST-----
   MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
   ...
   lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
   -----END CERTIFICATE REQUEST-----
   ```

   b. **The following command creates a CSR on the `host1` system:**

   ```
   # ikecert certlocal -kc -m 2048 -t rsa-sha384 \
   > -D "C=JA, O=Example1Co\, Inc., OU=JA-Example1x, CN=Example1x" \
   > -A "DN=C=JA, O=Example1Co\, Inc., OU=JA-Example1x"
   Creating software private keys.
   ...
   Finished successfully.
   -----BEGIN CERTIFICATE REQUEST-----
   MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
   ...
   8qlqdjaStLGfhDOO
   -----END CERTIFICATE REQUEST-----
   ```

2. **Submit the CSR to a CA.**

The CA can tell you how to submit the CSR. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your CSR into the form. When your request has been checked by the organization, the organization issues you signed certificates. For more information, see "Using Public Key Certificates in IKE" on page 132.

3. **Add each certificate to your system.**

   The `-a` option to the `ikecert certdb -a` adds the pasted object to the appropriate certificate database on your system. For more information, see "IKE With Public Key Certificates" on page 132.

   a. **Become an administrator.**

      For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*. If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

   b. **Add the public key and its certificate that you received from the CA.**

      ```
      # ikecert certdb -a < /tmp/PKIcert.eml
      ```

   c. **Add the CA's public certificate.**

      You might also need to add intermediate certificates.

      ```
      # ikecert certdb -a < /tmp/PKIca.eml
      ```

   d. **If the CA has sent a list of revoked certificates, add the CRL to the `certrldb` database:**

      ```
      # ikecert certrldb -a
      Press the Return key
      Paste the CRL
      -----BEGIN CRL-----
      ...
      -----END CRL----
      Press the Return key
      Press Control-D
      ```

4. **Use the `cert_root` keyword in the `/etc/inet/ike/config` file to identify the CA that issued the certificate.**

   Use the Distinguished Name (DN) of the CA's certificate.

a. **For example, the `ike/config` file on the `host2` system might appear similar to the following:**

```
# Trusted root cert
# This certificate is from Example CA
# This is the X.509 distinguished name for the CA's cert

cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example, CN=Example CA"

## Parameters that may also show up in rules.

p1_xform
 { auth_method rsa_sig oakley_group 14 auth_alg sha384 encr_alg aes}
p2_pfs 14

{
 label "US-host2 to JA-host1 - Example CA"
 local_id_type dn
 local_id  "C=US, O=Example2Co, OU=US-Example2m, CN=Example2m"
 remote_id "C=JA, O=Example1Co, OU=JA-Example1x, CN=Example1x"

 local_addr  192.0.2.213
 remote_addr 192.0.2.16

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

---

**Note -** All arguments to the `auth_method` parameter must be on the same line.

---

b. **On the `host1` system, create a similar file.**

Specifically, the `host1` `ike/config` file must do the following:

- Include the same `cert_root` value.
- Use `host1` values for local parameters.
- Use `host2` values for remote parameters.
- Create a unique value for the `label` keyword. This value must be different from the remote system's `label` value.

```
...
cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example, CN=Example CA"
...
{
 label "JA-host1 to US-host2 - Example CA"
 local_id_type dn
 local_id   "C=JA, O=Example1Co, OU=JA-Example1x, CN=Example1x"
 remote_id  "C=US, O=Example2Co, OU=US-Example2m, CN=Example2m"

 local_addr  192.0.2.16
 remote_addr 192.0.2.213
...
```

5. **Set the IKEv1 policies for handling revoked certificates.**

   Choose the appropriate option:

   - **No OCSP available**

     If the public key certificate provides a URI to reach the OCSP server but your system cannot connect to the Internet, add the keyword `ignore_ocsp` to the `ike/config` file.

     ```
     # Trusted root cert
     ...
     cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example,...
     ignore_ocsp
     ...
     ```

     The `ignore_ocsp` keyword tells IKEv1 to assume that the certificate is valid.

   - **No CRL available**

     If the CA does not provide a reliable source for CRLs or your system cannot connect to the Internet to retrieve CRLs, add the keyword `ignore_crls` to the `ike/config` file.

     ```
     # Trusted root cert
     ...
     cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example,...
     ignore_crls
     ...
     ```

   - **URI for CRLs or OCSP available**

     If the CA provides a central distribution point for revoked certificates, you can modify the `ike/config` file to use the URI.

     See "How to Handle Revoked Certificates in IKEv1" on page 185 for examples.

**Example 35** Using `rsa_encrypt` When Configuring IKEv1

When you use `auth_method rsa_encrypt` in the `ike/config` file, you must add the peer's certificate to the `publickeys` database.

1. Send the certificate to the remote system's administrator.

You can paste the certificate into an email message.

For example, the host2 administrator would send the following message:

```
To: admin@host1.ja.example.com
From: admin@host2.us.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MII...
----END X509 CERTIFICATE-----
```

The host1 administrator would send the following message:

```
To: admin@host2.us.example.com
From: admin@host1.ja.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MII
...
-----END X509 CERTIFICATE-----
```

2. On each system, add the emailed certificate to the local publickeys database.

```
# ikecert certdb -a < /tmp/saved.cert.eml
```

The authentication method for RSA encryption hides identities in IKE from eavesdroppers. Because the rsa_encrypt method hides the peer's identity, IKEv1 cannot retrieve the peer's certificate. As a result, the rsa_encrypt method requires that the IKEv1 peers know each other's public keys.

Therefore, when you use an auth_method of rsa_encrypt in the /etc/inet/ike/config file, you must add the peer's certificate to the publickeys database. The publickeys database then holds at least three certificates for each communicating pair of systems:

- Your public key certificate
- The CA's certificate chain
- The peer's public key certificate

**Troubleshooting –** The IKEv1 payload, which includes at least three certificates, can become too large for rsa_encrypt to encrypt. Errors such as "authorization failed" and "malformed payload" can indicate that the rsa_encrypt method cannot encrypt the total payload. Reduce the size of the payload by using a method, such as rsa_sig, that requires only two certificates.

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

## ▼ How to Handle Revoked Certificates in IKEv1

Revoked certificates are certificates that are compromised for some reason. A revoked certificate that is in use is a security risk. You have options when verifying certificate revocation. You can use a static list or you can verify revocations dynamically over the HTTP protocol. You have four ways to handle revoked certificates.

- You can instruct IKEv1 to ignore CRLs or OCSP whose uniform resource indicator (URI) is embedded in the certificate. This option is shown in Step 5.
- You can instruct IKEv1 to access the CRLs or OCSP from a URI whose address is embedded in the public key certificate from the CA.
- You can instruct IKEv1 to access the CRLs from an LDAP server whose DN (directory name) entry is embedded in the public key certificate from the CA.
- You can provide the CRL as an argument to the `ikecert certrldb` command. For an example, see Example 36, "Pasting a CRL Into the Local `certrldb` Database for IKEv1," on page 187.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

1. **Display the certificate that you received from the CA.**

   For information about the arguments to the `ikecert certdb` command, see the `ikecert(8)` man page.

   For example, the following certificate was issued by a company's PKI. Details have been altered.

```
# ikecert certdb -lv cert-protect.example.com
Certificate Slot Name: 0   Type: dsa-sha256
   (Private key in certlocal slot )
 Subject Name: <O=Example, CN=cert-protect.example.com>
 Issuer Name: <CN=ExampleCo CO (Cl B), O=Example>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2013 Sep 19th, 21:11:11 GMT
      Not Valid After:  2017 Sep 18th, 21:11:11 GMT
   Public Key Info:
      Public Modulus  (n) (2048 bits): C575A...A5
      Public Exponent (e) (  24 bits): 010001
   Extensions:
      Subject Alternative Names:
              DNS = cert-protect.example.com
      Key Usage: DigitalSignature KeyEncipherment
      [CRITICAL]
   CRL Distribution Points:
      Full Name:
         URI = #Ihttp://www.example.com/pki/pkismica.crl#i
         DN = <CN=ExampleCo CO (Cl B), O=Example>
      CRL Issuer:
      Authority Key ID:
      Key ID:              4F ... 6B
      SubjectKeyID:        A5 ... FD
      Certificate Policies
      Authority Information Access
```

Notice the `CRL Distribution Points` entry.

- The URI entry indicates that this organization's CRL is available on the web.
- The DN entry indicates that the CRL is available on an LDAP server. Once accessed by IKE, the CRL is cached for further use.

To access the CRL, you need to reach a distribution point.

2. **Choose one of the following methods to access the CRL from a central distribution point.**

   - **Use the URI.**

     Add the keyword `use_http` to the host's `/etc/inet/ike/config` file. For example, the `ike/config` file would appear similar to the following:

     ```
     # Use CRL or OCSP from organization's URI
     use_http
     ...
     ```

   - **Use a web proxy.**

     Add the keyword `proxy` to the `ike/config` file. The `proxy` keyword takes a URL as an argument, as in the following:

     ```
     # Use web proxy to reach CRLs or OCSP
     proxy "http://proxy1:8080"
     ```

   - **Use an LDAP server.**

     Name the LDAP server as an argument to the `ldap-list` keyword in the host's `/etc/inet/ike/config` file. Your organization provides the name of the LDAP server. The entry in the `ike/config` file would appear similar to the following:

     ```
     # Use CRL from organization's LDAP
     ldap-list "ldap1.example.com:389,ldap2.example.com"
     ```

```
...
```

IKE retrieves the CRL and caches the CRL until the certificate expires.

**Example 36** Pasting a CRL Into the Local `certrldb` Database for IKEv1

If the CA's CRL is not available from a central distribution point, you can add the CRL manually to the local `certrldb` database. Follow the CA's instructions for extracting the CRL into a file, then add the CRL to the database with the `ikecert certrldb -a` command.

```
# ikecert certrldb -a < ExampleCo.Cert.CRL
```

## Configuring IKEv1 for Mobile Systems

IPsec and IKE require a unique ID to identify source and destination. For off-site or mobile systems that do not have a unique IP address, you must use another ID type. ID types such as `DNS`, `DN`, or `email` can be used to uniquely identify a system.

Off-site or mobile systems that have unique IP addresses are still best configured with a different ID type. For example, if the systems attempt to connect to a central site from behind a NAT box, their unique addresses are not used. A NAT box assigns an arbitrary IP address, which the central system would not recognize.

Preshared keys also do not work well as an authentication mechanism for mobile systems, because preshared keys require fixed IP addresses. Self-signed certificates, or certificates from a CA enable mobile systems to communicate with the central site.

The following task map lists procedures to configure IKEv1 to handle systems that log in remotely to a central site.

**TABLE 12** Configuring IKEv1 for Mobile Systems Task Map

| Task | Description | For Instructions |
|---|---|---|
| Communicate with a central site from off-site. | Enables off-site systems to communicate with a central site. The off-site systems might be mobile. | "How to Configure IKEv1 for Off-Site Systems" on page 188 |
| Use a CA's public certificate and IKEv1 on a central system that accepts traffic from mobile systems. | Configures a gateway system to accept IPsec traffic from a system that does not have a fixed IP address. | Example 37, "Configuring a Central Computer That Uses IKEv1 to Accept Protected Traffic From a Mobile System," on page 191 |
| Use a CA's public certificate and IKEv1 on a system that does not have a fixed IP address. | Configures a mobile system to protect its traffic to a central site, such as company headquarters. | Example 38, "Configuring a System Behind a NAT With IPsec and IKEv1," on page 192 |

| Task | Description | For Instructions |
|---|---|---|
| Use self-signed certificates and IKEv1 on a central system that accepts traffic from mobile systems. | Configures a gateway system with self-signed certificates to accept IPsec traffic from a mobile system. | Example 39, "Accepting Self-Signed Certificates From a Mobile System," on page 193 |
| Use self-signed certificates and IKEv1 on a system that does not have a fixed IP address. | Configures a mobile system with self-signed certificates to protect its traffic to a central site. | Example 40, "Using Self-Signed Certificates to Contact a Central System," on page 194 |

## ▼ How to Configure IKEv1 for Off-Site Systems

**Before You Begin** You must assume the `root` role. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*. If you administer remotely, see Example 19, "Configuring IPsec Policy Remotely by Using an `ssh` Connection," on page 107 and "How to Remotely Administer ZFS With Secure Shell" in *Managing Secure Shell Access in Oracle Solaris 11.4* for secure remote login instructions.

1. **Configure the central system to recognize mobile systems.**

   a. **Configure the `ipsecinit.conf` file.**

   The central system needs a policy that allows a wide range of IP addresses. Later, certificates in the IKE policy ensure that the connecting systems are legitimate.

   ```
   # /etc/inet/ipsecinit.conf on central
   # Keep everyone out unless they use this IPsec policy:
   {} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}
   ```

   b. **Configure the IKEv1 configuration file.**

   DNS identifies the central system. Certificates are used to authenticate the system.

   ```
   ## /etc/inet/ike/ike.config on central
   # Global parameters
   #
   # Find CRLs by URI, URL, or LDAP
   # Use CRL from organization's URI
   use_http
   #
   # Use web proxy
   proxy "http://somecache.domain:port/"
   #
   # Use LDAP server
   ldap_server   "ldap-server1.example.org,ldap2.example.org:port"
   #
   # List CA-signed certificates
   ```

```
cert_root    "C=US, O=Domain Org, CN=Domain STATE"
#
# List self-signed certificates - trust server and enumerated others
#cert_trust    "DNS=central.example.org"
#cert_trust    "DNS=mobile.example.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust    "email=root@central.example.org"
#cert_trust    "email=user1@mobile.example.org"
#

# Rule for mobile systems with certificate
{
  label "Mobile systems with certificate"
  local_id_type DNS
# CA's public certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256 }
}
```

2. **Log in to each mobile system, and configure the system to find the central system.**

   a. **Configure the `/etc/hosts` file.**

   The `/etc/hosts` file does not need an address for the mobile system, but can provide one. The file must contain a public IP address for the central system, *central*.

   ```
   # /etc/hosts on mobile
   central 192.xxx.xxx.x
   ```

   b. **Configure the `ipsecinit.conf` file.**

   The mobile system needs to find the central system by its public IP address. The systems must configure the same IPsec policy.

   ```
   # /etc/inet/ipsecinit.conf on mobile
   # Find central
   {raddr 192.xxx.xxx.x} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}
   ```

   c. **Configure the IKEv1 configuration file.**

The identifier cannot be an IP address. The following identifiers are valid for mobile systems:

- `DN=`*ldap-directory-name*
- `DNS=`*domain-name-server-address*
- `email=`*email-address*

Certificates are used to authenticate the mobile system, *mobile*.

```
## /etc/inet/ike/ike.config on mobile
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server   "ldap-server1.example.org,ldap2.example.org:port"
#
# List CA-signed certificates
cert_root    "C=US, O=Domain Org, CN=Domain STATE"
#
# Self-signed certificates - trust me and enumerated others
#cert_trust    "DNS=mobile.example.org"
#cert_trust    "DNS=central.example.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust    "email=user1@example.org"
#cert_trust    "email=root@central.example.org"
#
# Rule for off-site systems with root certificate
{
 label "Off-site mobile with certificate"
 local_id_type DNS

# NAT-T can translate local_addr into any public IP address
# central knows me by my DNS

 local_id "mobile.example.org"
 local_addr 0.0.0.0/0

# Find central and trust the root certificate
 remote_id "central.example.org"
 remote_addr 192.xxx.xxx.x
p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256 }
}
```

3. **Enable the `ike:default` service.**

```
# svcadm enable svc:/network/ipsec/ike:default
```

**Example 37** Configuring a Central Computer That Uses IKEv1 to Accept Protected Traffic From a Mobile System

IKE can initiate negotiations from behind a NAT box. However, the ideal setup for IKE is without an intervening NAT box. In the following example, the CA's public certificate has been placed on the mobile system and the central system. A central system accepts IPsec negotiations from a system behind a NAT box. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 38, "Configuring a System Behind a NAT With IPsec and IKEv1," on page 192.

```
## /etc/hosts on main1
main1 192.0.2.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.example.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.example.org,ldap2.example.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExampleCA Inc, OU=CA-Example, CN=Example CA"
#
# Rule for off-site systems with root certificate
{
  label "Off-site system with root certificate"
  local_id_type DNS
  local_id "main1.example.org"
  local_addr 192.0.2.100

# CA's public certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256}
}
```

**Example 38** Configuring a System Behind a NAT With IPsec and IKEv1

In the following example, the CA's public certificate is placed on the mobile system and the central system. `mobile1` is connecting to the company headquarters from home. The Internet service provider (ISP) network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. For setting up the computer at company headquarters, see Example 37, "Configuring a Central Computer That Uses IKEv1 to Accept Protected Traffic From a Mobile System," on page 191.

```
## /etc/hosts on mobile1
mobile1 192.0.2.3
main1 192.0.2.100

## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.0.2.100} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}

## /etc/inet/ike/ike.config on mobile1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.example.org:8080/"
#
```

```
# Use LDAP server
ldap_server   "ldap1.example.org,ldap2.example.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExampleCA Inc, OU=CA-Example, CN=Example CA"
#
# Rule for off-site systems with root certificate
{
  label "Off-site mobile1 with root certificate"
  local_id_type DNS
  local_id "mobile1.example.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the root certificate
  remote_id "main1.example.org"
  remote_addr 192.0.2.100

p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256 }
}
```

**Example 39** Accepting Self-Signed Certificates From a Mobile System

In the following example, self-signed certificates have been issued and are on the mobile and the central system. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 40, "Using Self-Signed Certificates to Contact a Central System," on page 194.

```
## /etc/hosts on main1
main1 192.0.2.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Self-signed certificates - trust me and enumerated others
cert_trust    "DNS=main1.example.org"
cert_trust    "jdoe@example.org"
cert_trust    "user2@example.org"
cert_trust    "user3@example.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site systems with trusted certificates"
  local_id_type DNS
  local_id "main1.example.org"
  local_addr 192.0.2.100

# Trust the self-signed certificates
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256 }
}
```

**Example 40** Using Self-Signed Certificates to Contact a Central System

In the following example, `mobile1` is connecting to the company headquarters from home. The certificates have been issued and placed on the mobile and the central system. The ISP network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. To set up the computer at company headquarters, see Example 39, "Accepting Self-Signed Certificates From a Mobile System," on page 193.

```
## /etc/hosts on mobile1
mobile1 192.0.2.3
main1 192.0.2.100
```

```
## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.0.2.100} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}
```

```
## /etc/inet/ike/ike.config on mobile1
# Global parameters

# Self-signed certificates - trust me and the central system
cert_trust    "jdoe@example.org"
cert_trust    "DNS=main1.example.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site mobile1 with trusted certificate"
  local_id_type email
  local_id "jdoe@example.org"
  local_addr 0.0.0.0/0
# Find main1 and trust the certificate
  remote_id "main1.example.org"
  remote_addr 192.0.2.100

p2_pfs 14

p1_xform
{auth_method rsa_sig oakley_group 14 encr_alg aes auth_alg sha256 }
}
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see "Protecting a VPN With IPsec" on page 112. For other examples of IPsec policy, see "How to Secure Network Traffic Between Two Servers With IPsec" on page 105.

♦ ♦ ♦ CHAPTER 11

# Troubleshooting IPsec and Its Key Management Services

This chapter describes how to troubleshoot IPsec and its keys, how to view configuration information, and how to view information about active IPsec, IKE, and manual key services.

The chapter contains the following information:

## Troubleshooting IPsec and Its Key Management Configuration

You can set up your system for troubleshooting before or during a problem that needs troubleshooting.

When troubleshooting, you can run many commands in a profile shell as an administrator with the Network IPsec Management rights profile. However, to read logs you must assume the `root` role.

The prompts in the troubleshooting sections indicate whether you must have rights to run a command.

- `#` prompt – A user with the appropriate administrative rights or a role with those rights can run the command.
- `%` prompt – A regular user can run the command.

## ▼ How to Prepare IPsec and IKE Systems for Troubleshooting

Before you enable IPsec and its key management services, you can set up your system with logs and tools that aid in troubleshooting.

1. **Locate the logs for the IPsec and IKEv2 services.**

   The `-L` option provides the full path to the logs. These logs contain information messages as well as error messages.

   ```
   $ svcs -L policy
   /var/svc/log/network-ipsec-policy:default.log

   $ svcs -L ikev2
   /var/svc/log/network-ipsec-ike:ikev2.log
   ```

2. **Configure a debug log file for IKEv2.**

   The `root` role can read these logs.

   ```
   $ svcprop ikev2 | grep debug
   config/debug_level              astring     op
   config/debug_logfile            astring     /var/log/ikev2/in.ikev2d.log
   ```

   The levels of debugging are described in the `ikeadm(8)` man page. The values `verbose` and `all` are useful when troubleshooting.

3. **(Optional) Configure the debug level.**

   The following command sets the debug level permanently. To set the debug level temporarily, see Example 43, "Setting a New Debug Level on a Running IKE Daemon," on page 204.

   ```
   $ pfbash svccfg -s ikev2 setprop config/debug_level = all
   ```

   If the `ikev2` service is enabled, it must be refreshed to use the new debug level.

   ```
   $ svcadm refresh ikev2
   ```

4. **(Optional) Install the `wireshark` package.**

   The Wireshark application can read `snoop` output.

   ```
   $ pkg info -r wireshark
             Name: diagnostic/wireshark
          Summary: Graphical network protocol analyzer
         Category: Applications/Internet
            State: Not installed
   ```

```
     Publisher: solaris
...
          FMRI: pkg://solaris/diagnostic/wireshark@version
# pkg install diagnostic/wireshark
```

## ▼ How to Troubleshoot Systems Before IPsec and IKE Are Running

You can check the syntax of the IPsec configuration file, the IPsec keys file, and the validity of certificates in the keystore before running the services.

1. **Verify the syntax of the IPsec configuration file.**

   ```
   # ipsecconf -c /etc/inet/ipsecinit.conf
   ipsecconf: Invalid pattern on line 5: ukp
   ipsecconf: form_ipsec_conf error
   ipsecconf: Malformed command (fatal):
   { ukp 58 type 133-137 dir out} pass {}

   ipsecconf: 1 policy rule(s) contained errors.
   ipsecconf: Fatal error - exiting.
   ```

   If the output shows an error, fix it and run the command until the verification succeeds.

2. **Verify the syntax of the `ipseckeys` file.**

   ```
   # ipseckey -c /etc/inet/secret/ipseckeys
   Config file /etc/inet/secret/ipseckeys has insecure permissions,
   will be rejected in permanent config.
   ```

   If the output shows an error, fix the error then refresh the service.

   ```
   # svcadm refresh ipsec/policy
   ```

   **Note -** The IKE configuration files and IKE preshared key files are validated by a running IKE daemon.

3. **Verify the validity of the certificates.**

   - To verify the validity of self-signed certificates in IKEv2, perform Step 4 in "How to Configure IKEv2 With Self-Signed Public Key Certificates" on page 152.
   - To verify that a public key certificate is not revoked in IKEv2, follow the procedure "How to Set a Certificate Validation Policy in IKEv2" on page 161.

- To verify the validity of self-signed certificates in IKEv1, perform Step 4 in "How to Configure IKEv1 With Self-Signed Public Key Certificates" on page 174.
- To verify that a public key certificate is not revoked in IKEv1, follow the procedure "How to Handle Revoked Certificates in IKEv1" on page 185.

**Next Steps** If your configuration does not work when you enable IPsec and its keying services, you must troubleshoot while the services are running.

## ▼ How to Troubleshoot Systems When IPsec Is Running

On running systems that are exchanging or attempting to exchange packets by using IKE, you can use the `ikeadm` command to view statistics, rules, preshared keys and other things. You can also use the log files and selected tools, such as the Wireshark application.

1. **Investigate the following items:**

   - **Verify that the `policy` and appropriate key management services are enabled.**

     On the following test system, the `manual-key` service is being used for key management:

     ```
     $ svcs -a  | grep ipsec
     online         Feb_04   svc:/network/ipsec/manual-key:default
     online         Feb_04   svc:/network/ipsec/ipsecalgs:default
     online         Feb_04   svc:/network/ipsec/policy:default
     disabled       Feb_28   svc:/network/ipsec/ike:ikev2
     disabled       Feb_28   svc:/network/ipsec/ike:default
     ```

     If the service is disabled, enable it.

     You can use both IKE services concurrently. You can also use manual keys and IKE concurrently, but this configuration could result in oddities that are difficult to troubleshoot.

   - **View the end of the log file for the IKEv2 service.**

     ```
     $ svcs -xL ikev2
     svc:/network/ipsec/ike:ikev2 (IKEv2 daemon)
      State: disabled since October  10, 2013 10:10:40 PM PDT
     Reason: Disabled by an administrator.
        See: http://support.oracle.com/msg/SMF-8000-05
        See: in.ikev2d(8)
     ```

```
   See: /var/svc/log/network-ipsec-ike:ikev2.log
Impact: This service is not running.
   Log:
Oct 01 13:20:20: (1)  Property "debug_level" set to: "op"
Oct 01 13:20:20: (1)  Errors and debug messages will be written to:
                        /var/log/ikev2/in.ikev2d.log
[ Oct 10 10:10:10 Method "start" exited with status 0. ]
[ Oct 10 10:10:40 Stopping because service disabled. ]
[ Oct 10 10:10:40 Executing stop method (:kill). ]

   Use: 'svcs -Lv svc:/network/ipsec/ike:ikev2' to view the complete log.
```

- **(Optional) You can set a temporary value for the debug level of the running daemon.**

  ```
  # ikeadm set debug verbose /var/log/ikev2/in.ikev2d.log
  Successfully changed debug level from 0x80000000 to 0x6204
  Debug categories enabled:
          Operational / Errors
          Config file processing
          Interaction with Audit
          Verbose Operational
  ```

2. **Verify that the output of the `ipsecconf` command matches the contents of the policy file.**

```
# ipsecconf
#INDEX 14
...
{  laddr 192.0.2.12 raddr 192.0.2.17 }
 ipsec   { encr_algs aes(256) encr_auth_algs sha512 sa shared }
...
{  laddr 192.0.2.66 raddr 192.0.2.77 }
 ipsec   { encr_algs aes(256) encr_auth_algs sha512 sa shared }

# cat /etc/inet/ipsecinit.conf
...
{  laddr 192.0.2.12 raddr 192.0.2.17 }
 ipsec   { encr_algs aes(256) encr_auth_algs sha512 sa shared }

{  laddr 192.0.2.66 raddr 192.0.2.77 }
 ipsec   { encr_algs aes(256) encr_auth_algs sha512 sa shared }
```

**Note -** Wildcard addresses can obscure a match, so verify that any specific addresses in the `ipsecinit.conf` file are within the range of wildcard addresses in the output of `ipsecconf`.

If no output prints for the `ipsecconf` command, verify that the policy service is enabled and refresh the service.

```
$ svcs policy
STATE          STIME    FMRI
online         Apr_10   svc:/network/ipsec/policy:default
```

If the output shows an error, edit the `ipsecinit.conf` file to fix the error then refresh the service.

3. **Validate your IKEv2 configuration.**

   For configuration output that might require fixing, see Example 41, "Fixing an Invalid IKEv2 Configuration," on page 203 and Example 42, "Fixing a `No matching IKEv2 rule` Issue," on page 203. The output in the following example indicates that the configuration is valid.

```
# /usr/lib/inet/in.ikev2d -c
Feb 04 12:08:25: (1)    Reading service properties from smf(5) repository.
Feb 04 12:08:25: (1)    Property "config_file" set to: "/etc/inet/ike/ikev2.config"
Feb 04 12:08:25: (1)    Property "debug_level" set to: "all"
Feb 04 12:08:25: (1)    Warning: debug output being written to stdout.
Feb 04 12:08:25: (1)    Checking IKE rule #1: "Test 104 to 113"
Feb 04 12:08:25: (1)    Configuration file /etc/inet/ike/ikev2.config is valid.
Feb 04 12:08:25: (1)    Pre-shared key file /etc/inet/ike/ikev2.preshared is valid.
```

---

**Note -** The warning about debug output does not change even after you specify a debug log file. If you specify a value for the `debug_logfile` service property, the warning means that debug output is being delivered to that file. Otherwise, debug output is delivered to the console.

---

- In the `Checking IKE rule` lines, verify that the IKE rules connect the appropriate IP addresses. For example, the following entries match. The `laddr` value from the `ipsecinit.conf` file matches the `local_addr` value from the `ikev2.config` file, and the remote addresses match.

```
{  laddr 192.0.2.84 raddr 192.0.2.73 }      /** ipsecinit.conf **/
                 ipsec {encr_algs aes encr_auth_algs sha512 sa shared}

 local_addr   192.0.2.84                         /** ikev2.config **/
 remote_addr  192.0.2.73                         /** ikev2.config **/
```

  If the entries do not correspond, fix the configuration to identify the correct IP addresses.

---

**Note -** Rules can have wildcard addresses such as `192.0.2.32/27` that cover a range of addresses. Verify the range against specific addresses.

---

- If the `Pre-shared key file` line indicates that the file is not valid, fix the file.

  Check for typographical errors. Also, in IKEv2, check that the label value in the rule in `ikev2.config` matches the label value in the `ikev2.preshared` file. Then, if you are using two keys, verify that the local preshared key on one system matches the remote preshared key on its peer, and that the remote key matches the local key on the peer.

  If your configuration still does not work, see "Troubleshooting IPsec and IKE Semantic Errors" on page 205.

**Example 41** Fixing an Invalid IKEv2 Configuration

In the following output, the lifetime of the IKE SA is too short.

```
# /usr/lib/inet/in.ikev2d -c
...
May 08 08:52:49: (1) WARNING: Problem in rule "Test 104 to 113"
May 08 08:52:49: (1)  HARD lifetime too small (60 < 100)
May 08 08:52:49: (1)   -> Using 100 seconds (minimum)
May 08 08:52:49: (1) Checking IKE rule #1: "config 192.0.2.73 to 192.0.2.84"
...
```

This value has been explicitly set in the `ikev2.config` file. To remove the warning, change the lifetime value to at least `100` and refresh the service.

```
# pfedit /etc/inet/ike/ikev2.config
...
## childsa_lifetime_secs   60
childsa_lifetime_secs   100
...
# /usr/lib/inet/in.ikev2d -c
...
# svcadm refresh ikev2
```

**Example 42** Fixing a `No matching IKEv2 rule` Issue

In the following output, a preshared key is defined but is not used in a rule.

```
# /usr/lib/inet/in.ikev2d -c
Feb 4 12:58:31: (1)  Reading service properties from smf(5) repository.
Feb 4 12:58:31: (1)  Property "config_file" set to: "/etc/inet/ike/ikev2.config"
Feb 4 12:58:31: (1)  Property "debug_level" set to: "op"
Feb 4 12:58:31: (1)  Warning: debug output being written to stdout.
Feb 4 12:58:31: (1)  Checking IKE rule #1: "Test 104 to 113"
Feb 4 12:58:31: (1)  Configuration file /etc/inet/ike/ikev2.config is valid.
Feb 4 12:58:31: (1)  No matching IKEv2 rule for pre-shared key ending on line 12
Feb 4 12:58:31: (1)  Pre-shared key file /etc/inet/ike/ikev2.preshared is valid.
```

The output indicates that only one rule exists.

- If the rule requires a preshared key, then the label of the preshared key does not match the label of the rule. Fix the `ikev2.config` rule label and the `ikev2.preshared` key label to match.
- If the rule uses a certificate, then you can remove or comment out the preshared key that ends on line 12 in the `ikev2.preshared` file to prevent the `No matching` message.

**Example 43** Setting a New Debug Level on a Running IKE Daemon

In the following output, debug output is set to `all` in the `ikev2` service.

```
# /usr/lib/inet/in.ikev2d -c
Feb 4 12:58:31: (1)  Reading service properties from smf(5) repository.
...
Feb 4 12:58:31: (1)  Property "debug_level" set to: "all"
...
```

If you have completed Step 2 in "How to Troubleshoot Systems Before IPsec and IKE Are Running" on page 199 and the debug output is still `op` rather than `all`, use the `ikeadm` command to set the debug level on the running IKE daemon.

```
# ikeadm set debug_level all
```

**Example 44** Preventing the Loss of IKEv2 Messages From Intermediate Devices

Because intermediate devices are dropping IKEv2 messages, the administrator lowers the `fragmentation_mtu` value of the `ikev2` service.

1. The administrator displays the values of the fragmentation properties.

   ```
   $ svcprop ikev2 | grep fragment
   config/fragmentation_enable boolean true
   config/fragmentation_mtu integer 1350
   ```
2. After determining the path MTU (maximum transmission unit) between the hosts exchanging IKEv2 packets, the administrator uses that value as the `fragmentation_mtu` value.

   The path MTU is 1330.

   ```
   $ pfbash svccfg -s ike:ikev2 setprop config/fragmentation_mtu = 1330
   # svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
   ```

3. The administrator refreshes and restarts the service, then verifies the fragmentation value.

```
$ svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
$ svcprop ikev2 | grep fragment
config/fragmentation_enable boolean true
config/fragmentation_mtu integer 1330
```

## Troubleshooting IPsec and IKE Semantic Errors

If the investigations in "How to Troubleshoot Systems When IPsec Is Running" on page 200 fail to handle the problem, then the semantics of your configuration is the likely problem, rather than the syntax of your files or the service configuration.

- If both the `ike:default` and `ike:ikev2` service instances are enabled, ensure that the IKEv2 and IKEv1 rules do not overlap. Rules that apply to the same network endpoints can result in redundant IPsec SAs and could cause a lack of connectivity in certain situations.

  If you change an IKE rule, read the rule into the kernel.

  ```
  # ikeadm -v[1|2] read rule
  ```
- If you are running IKEv1, make sure that the algorithm mechanisms in your rules are available on the IKEv1 system that you are connecting to. To view the available algorithms, run the `ikeadm dump` *algorithms* command on the system that does not support IKEv2:

  ```
  # ikeadm dump groups      Available Diffie-Hellman groups
  # ikeadm dump encralgs    All IKE encryption algorithms
  # ikeadm dump authalgs    All IKE authentication algorithms
  ```

  Correct both the IPsec and IKEv1 policy files to use algorithms that are available on both systems. Then, restart the IKEv1 service and refresh the IPsec service.

  ```
  # svcadm restart ike:default; svcadm refresh ipsec/policy
  ```
- If you are using preshared keys with IKEv1, and the remote IKEv1 system is rebooted, run the `ipseckey flush` command on the local system.
- If you are using self-signed certificates, verify with the other administrator that a certificate with the same DN has not been re-created and that the hash values of your certificates match. For the verification steps, see Step 4 in "How to Configure IKEv2 With Self-Signed Public Key Certificates" on page 152.

  If the certificate is updated, import the new certificate, then refresh and restart the IKEv2 service.
- Use the `ikeadm -v2 dump | get` command to view the current IKEv2 configuration. For a usage summary, see "Viewing IKE Information" on page 207.

- Use the `kstat2` command to display IPsec-related statistics. The `-p` option displays parseable output and the `-h` option displays human-readable output. For more information, see the `kstat2(8)` man page.

  ```
  $ pfbash kstat2 -p /net/ipsecesp/esp_stat/0
  $ kstat2 -p /net/ipsecah/ah_stat/0
  $ kstat2 -p /net/ip/ipsec_stat/0
  ```

  The `kstat2` output in the following example indicates no problems in the `ipsecesp` module.

  ```
  $ kstat2 -h /net/ipsecesp/esp_stat/0
  kstat:/net/ipsecesp/esp_stat/0
          acquire_requests                12
          bad_auth                        0
          bad_decrypt                     0
          bad_padding                     0
          bytes_expired                   0
          crtime                          541DT16H14M00S from boot
          crypto_async                    0
          crypto_failures                 0
          crypto_sync                     3.48 M
          good_auth                       1.62 M
          keysock_in                      7.05 k
          num_aalgs                       10
          num_ealgs                       14
          out_discards                    0
          out_requests                    1.86 M
          replay_early_failures           0
          replay_failures                 0
          sa_port_renumbers               0
          snaptime                        547DT16H09M38S from boot
          tfc_dummy_pkts                  0
          tfc_pad_pkts                    0
  ```

- Use the `snoop` command to view the traffic that is not being protected. The Wireshark application can read `snoop` output. For an example of `snoop` output, see "How to Verify That Packets Are Protected With IPsec" on page 127.

## Viewing Information About IPsec and Its Keying Services

---

**Note -** For most commands, you must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

---

### Viewing IPsec and Manual Key Service Properties

You can view the name of the IPsec policy file and the file that holds manual keys.

- To show the name of the IPsec configuration file:

  ```
  $ svcprop policy:default | grep config_file
  config/config_file astring /etc/inet/ipsecinit.conf
  ```
- To show the name of the file that holds manual keys for IPsec:

  ```
  $ svcprop manual-key | grep ^config/config_file
  config/config_file astring /etc/inet/secret/ipseckeys
  ```

### Viewing IKE Information

You can view the properties of the IKE service, aspects of the IKE state and IKE daemon object, and certificate validation policy. If you are running both IKE services, you can display information per service or for both services. These commands can be helpful during testing, troubleshooting, and monitoring.

- Viewing the properties of the IKE service instances – The output displays the configurable properties of the IKEv2 service, including the names of the configuration files.

  ---

  **Note -** Review the `ipsecconf(8)`. `in.ikev2d(8)`, and `in.iked(8)` man pages to ensure that you can or should modify a property in the `config` group of the IPsec, IKEv2, or IKEv1 service. For example, IKEv2 configuration files are created with special permissions and owned by `ikeuser`. The permissions and file owner must not be changed.

  ---

  ```
  $ svcprop ipsec/ike:ikev2 | grep ^config
  config/allow_keydump boolean false
  ```

```
config/config_file astring /etc/inet/ike/ikev2.config
config/debug_level astring op
config/debug_logfile astring /var/log/ikev2/in.ikev2d.log
config/fragmentation_enable boolean true
config/fragmentation_mtu integer 1350
config/http_cert_cache_dir astring /var/user/ikeuser/certcache
config/ignore_errors boolean false
config/kmf_policy astring /etc/inet/ike/kmf-policy.xml
config/max_child_sas integer 0
config/max_threads integer 0
config/min_threads integer 0
config/preshared_file astring /etc/inet/ike/ikev2.preshared
config/response_wait_time integer 30
config/value_authorization astring solaris.smf.value.ipsec
config/admin_privilege astring base
```

The output in the following example displays the configurable properties of the IKEv1 service.

```
$ svcprop ipsec/ike:default | grep ^config
config/admin_privilege astring base
config/config_file astring /etc/inet/ike/config
config/debug_level astring op
config/debug_logfile astring /var/log/in.iked.log
config/ignore_errors boolean false
config/max_listeners integer 2048
config/preshared_file astring /etc/inet/secret/ike.preshared
config/value_authorization astring solaris.smf.value.ipsec
```

- Viewing the current state of the IKE daemon – The output in the following example displays the arguments to the `ikeadm` command. These arguments display the current state of the daemon.

---

**Note -** To use the `ikeadm` command, the IKE daemon must be running.

---

```
$ ikeadm help
...
        get   debug|priv|stats|p1|ikesa|rule|preshared|defaults [identifier]
        dump  p1|ikesa|rule|preshared|certcache|groups|encralgs|authalgs
        read  rule|preshared [filename]
        help  [get|set|add|del|dump|flush|read|write|token|help]
```

- Showing the syntax of a specific argument to the `ikeadm` command – Use the `help` subcommands to show command argument syntax. For example:

```
$ ikeadm help read
This command reads a new configuration file into
in.iked, discarding the old configuration info.

Sets of data that may be read include:
        rule            all phase 1/ikesa rules
        preshared       all preshared keys

A filename may be provided to specify a source file
other than the default.
```

- Viewing preshared keys – You can view preshared keys for IKEv1 and IKEv2.

---

**Note -** If you are running only one IKE version, you can omit the -v option.

---

For IKEv2:

```
# ikeadm -v2 dump preshared
```

For IKEv1:

```
# ikeadm set priv keymat
# ikeadm -v1 dump preshared

PSKEY: Rule label: "Test PSK 197 to 56"
PSKEY: Local pre-shared key (80 bytes): 74206272696c6c696720...3/584
PSKEY: Remote pre-shared key (80 bytes): 74206272696c6c696720...3/584

Completed dump of preshared keys
```

- Viewing IKE SAs – The output includes information about the SA, the transform, the local and remote systems, and other details. If communication has not been requested, no SAs exist, so no information exists to display.

```
# ikeadm -v2 dump ikesa
IKESA: SPIs: Local 0xd3db95689459cca4  Remote 0xb5878717f5cfa877
...
XFORM: Encryption alg: aes-cbc(256..256); Authentication alg: hmac-sha512
...
LOCIP: AF_INET: port 500, 192.0.2.68 (example-3).
...
REMIP: AF_INET: port 500, 192.0.2.67 (ex-2).
...
```

```
LIFTM: SA expires in 11459 seconds (3.18 hours)
...
STATS: 0 IKE SA rekeys since initial AUTH.
LOCID: Initiator identity, type FQDN
...
CHILD: ESP Inbound SPI: 0x94841ca3, Outbound SPI 0x074ae1e5
...
Completed dump of IKE SA info
```

- Viewing active IKE rules – A listed IKE rule might not be in use, but it is available for use.

```
# ikeadm -v2 dump rule

GLOBL: Label 'Test Rule1 for PSK', key manager cookie 1
GLOBL: Local auth method=pre-shared key
GLOBL: Remote auth method=pre-shared key

GLOBL: childsa_pfs=false
GLOBL: authentication_lifetime=86400 seconds (1.00 day)
GLOBL: childsa_lifetime=120 seconds (2.00 minutes)
GLOBL: childsa_softlife=108 seconds (1.80 minute)
GLOBL: childsa_idletime=60 seconds
GLOBL: childsa_lifetime_kb=122880 kilobytes (120.00 MB)
GLOBL: childsa_softlife_kb=110592 kilobytes (108.00 MB)
LOCIP: IP address range(s):
LOCIP: 192.0.2.66
REMIP: IP address range(s):
REMIP: 192.0.2.12
LOCID: Identity descriptors:
LOCID: Includes:
LOCID:       fqdn="gloria@ms.mag"
REMID: Identity descriptors:
REMID: Includes:
REMID:       fqdn="gloria@ms.mag"
XFRMS: Available Transforms:

XF  0: Encryption alg: aes-cbc(128..256); Authentication alg: hmac-sha512
XF  0: PRF: hmac-sha512 ; Diffie-Hellman Group: 2048-bit MODP (group 14)
XF  0: IKE SA lifetime before rekey: 14400 seconds (4.00 hours)

Completed dump of policy rules
```

- Viewing certificate validation policy in IKEv2 – You must specify the `dbfile` value and the `policy` value.

- Dynamically downloaded CRLs might require administrator intervention to adjust the responder timeout.

  In the output in the following example, the CRLs are downloaded from the URI that is embedded in the certificate, then the lists are cached. When the cache contains an expired CRL, a new CRL is downloaded to replace the old one.

  ```
  # kmfcfg list dbfile=/etc/inet/ike/kmf-policy.xml policy=default
  …
  Validation Policy Information:
      Maximum Certificate Revocation Responder Timeout: 10
      Ignore Certificate Revocation Responder Timeout: true
  …
      CRL:
          Base filename: [not set]
          Directory: /var/user/ikeuser/crls
          Download and cache CRL: true
          CRL specific proxy override: www-proxy.cagate.example.com:80
          Ignore CRL signature: false
          Ignore CRL validity date: false
  IPsec policy bypass on outgoing connections: true
  …
  ```
- Statically downloaded CRLs require frequent administrator attention.

  When the administrator sets the CRL entries to the following values, the administrator is responsible for manually downloading the CRLs, populating the directory, and maintaining current CRLs:

  ```
  …
          Directory: /var/user/ikeuser/crls
          Download and cache CRL: false
          Proxy: [not set]
  …
  ```

## Managing IPsec and Its Keying Services

IPsec policy is enabled by default, but it lacks configuration information.

Key management is not enabled by default. You can configure IKE or manual key management, or both. Each IKE rule indicates which key management service is used. The `ikeadm` command can modify the running IKE daemon.

## Configuring and Managing IPsec and Its Keying Services

- Configuring and refreshing IPsec, then viewing policy:

  ```
  # pfedit /etc/inet/ipsecinit.conf
  # ipsecconf -c /etc/inet/ipsecinit.conf
  # svcadm refresh ipsec/policy
  # ipsecconf -Ln
  ```
- Configuring and enabling manual keys for IPsec:

  ```
  # pfedit -s /etc/inet/secret/ipseckeys
  # svcadm enable ipsec/manual-key
  ```
- Configuring and enabling IKEv2:

  ```
  # pfedit /etc/inet/ike/ikev2.config
  # /usr/lib/inet/in.ikev2d -c
  # svcadm enable ipsec/ike:ikev2
  ```
- Configuring and enabling IKEv1:

  ```
  # pfedit /etc/inet/ike/config
  # /usr/lib/inet/in.iked -c
  # svcadm enable ipsec/ike:default
  ```
- Verifying that IPsec and IKE are configured on a system where the services are enabled:

  ```
  # ipsecconf -Ln
  # ikeadm -v2 dump rule
  # ikeadm set priv keymat
  # ikeadm -v1 dump rule
  ```
- Modifying key management:

  For IKEv2:

  ```
  # pfedit /etc/inet/ike/ikev2.config
  # /usr/lib/inet/in.ikev2d -c
  # svcadm restart ipsec/ike:ikev2
  ```

  For IKEv1:

  ```
  # pfedit /etc/inet/ike/config
  # /usr/lib/inet/in.iked -c
  # svcadm restart ipsec/ike:default
  ```

  For manual key management:

```
# pfedit -s /etc/inet/secret/ipseckeys
# ipseckey -c /etc/inet/secret/ipseckeys
# svcadm refresh ipsec/manual-key
```

- Modifying IPsec and IKE configurable properties:

  IPsec service:

  ```
  # svccfg -s ipsec/policy setprop config/property = value
  # svcadm refresh ipsec/policy; svcadm restart ipsec/policy
  ```

  IKEv2 service for sensitive keying material:

  ```
  # svccfg -s ike:ikev2 editprop
  # svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
  ```

  IKEv2 service for other properties:

  ```
  # svccfg -s ike:ikev2 setprop config/property = value
  # svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
  ```

  For an example of why you would change a property value, see Example 44, "Preventing the Loss of IKEv2 Messages From Intermediate Devices," on page 204.

  IKEv1 service:

  ```
  # svccfg -s ipsec/ike setprop config/property = value
  # svcadm refresh ipsec/ike:default; svcadm restart ipsec/ike:default
  ```

  Manual keys service:

  ```
  # svccfg -s ipsec/manual-key setprop config/property = value
  # svcadm refresh ipsec/manual-key; svcadm restart ipsec/manual-key
  ```

- Configuring preshared keys for IKEv2:

  ```
  # pfedit -s /etc/inet/ike/ikev2.preshared
  # /usr/lib/inet/in.ikev2d -c
  # svcadm restart ikev2
  ```

- Configuring preshared keys for IKEv1:

  ```
  # pfedit -s /etc/inet/secret/ike.preshared
  # svcadm restart ike
  ```

## Managing the Running IKE Daemons

For more information, review the `ikeadm`(8) man page. The commands in this section are available only when the IKEv2 or IKEv1 daemon is running.

- Modifying the running IKE daemon:

  The following output displays the arguments to the `ikeadm` command that can modify the current state of the daemon. Some arguments are specific to the IKEv2 or the IKEv1 daemon.

```
$ ikeadm help
...
        set   priv level
        set   debug level [filename]
        add   rule|preshared {definition}|filename
        del   p1|ikesa|rule|preshared identifier
        flush p1|ikesa|certcache
        write rule|preshared filename
        token login|logout PKCS#11-Token-Object
```

- Showing the syntax of a specific argument to the `ikeadm` command:

```
$ ikeadm help add
This command adds items to in.iked's tables.

Objects that may be set include:
        rule            a phase 1 or IKE SA policy rule
        preshared       a preshared key

Objects may be entered on the command-line, as a
series of keywords and tokens contained in curly
braces ('{', '}'); or the name of a file containing
the object definition may be provided.

For security purposes, preshared keys may only be
entered on the command-line if ikeadm is running in
interactive mode.
```

- Modifying the IKEv2 daemon with the `ikeadm` command:

```
# ikeadm add rule | preshared {definition} | filename
# ikeadm flush ikesa
# ikeadm del ikesa | rule | preshared identifier
# ikeadm set debug level
# ikeadm token login | logout PKCS#11-Token-Object
# ikeadm write rule | preshared filename
```

- Modifying the IKEv1 daemon with the `ikeadm` command:

```
# ikeadm set debug level
# ikeadm set privlevel
# ikeadm add rule | preshared {definition} | filename
# ikeadm del p1 | rule | preshared identifier
# ikeadm flush p1 | certcache
# ikeadm del rule | preshared id
# ikeadm write rule | preshared filename
```

♦ ♦ ♦ C H A P T E R 1 2

# IPsec and Key Management Reference

This chapter contains reference information about IPsec, IKEv2, and IKEv1.

- "IPsec Reference" on page 217
- "IKEv2 Reference" on page 223
- "IKEv1 Reference" on page 227

For instructions on how to implement IPsec on your network, see Chapter 7, "Configuring IPsec". For an overview of IPsec, see Chapter 6, "About IP Security Architecture".

For instructions on implementing IKE, see Chapter 9, "Configuring IKEv2". For overview information, see Chapter 8, "About Internet Key Exchange".

## IPsec Reference

### IPsec Services, Files, and Commands

This section lists the IPsec services, selected IPsec RFCs, and the files and commands that are relevant to IPsec.

#### IPsec Services

The Service Management Facility (SMF) provides the following services for IPsec:

- `svc:/network/ipsec/policy` service – Manages IPsec policy. By default, this service is enabled. The value of the `config_file` property determines the location of the `ipsecinit.conf` file. The initial value is `/etc/inet/ipsecinit.conf`.
- `svc:/network/ipsec/ipsecalgs` service – Manages the algorithms that are available to IPsec. By default, this service is enabled.

- `svc:/network/ipsec/manual-key` service – Activates manual key management. By default, this service is disabled. The value of the `config_file` property determines the location of the `ipseckeys` configuration file. The initial value is `/etc/inet/secret/ipseckeys`.
- `svc:/network/ipsec/ike` service – Manages IKE. By default, this service is disabled. For the configurable properties, see "IKEv2 Service" on page 224 and "IKEv1 Service" on page 228.

For information about SMF, see Chapter 1, "Introduction to the Service Management Facility" in *Managing System Services in Oracle Solaris 11.4*. Also see the `smf`(7), `svcadm`(8), and `svccfg`(8) man pages.

## `ipsecconf` Command

You use the `ipsecconf` command to configure the IPsec policy for a host. When you run the command to configure the policy, the system creates the IPsec policy entries in the kernel. The system uses these entries to check the policy on all inbound and outbound IP packets. Packets that are not tunneled and forwarded are not subjected to policy checks that are added by using this command. The `ipsecconf` command also manages the IPsec entries in the security policy database (SPD). For IPsec policy options, see the `ipsecconf`(8) man page.

You must assume the `root` role to invoke the `ipsecconf` command. The command can configure entries that protect traffic in both directions. The command also can configure entries that protect traffic in only one direction.

Policy entries with a format of local address and remote address can protect traffic in both directions with a single policy entry. For example, entries that contain the patterns `laddr host1` and `raddr host2` protect traffic in both directions if no direction is specified for the named host. Thus, you need only one policy entry for each host.

Policy entries that are added by the `ipsecconf` command are not persistent over a system reboot. To ensure that the IPsec policy is active when the system boots, add the policy entries to the `/etc/inet/ipsecinit.conf` file, then refresh or enable the `policy` service. For examples, see "Protecting Network Traffic With IPsec" on page 103.

## `ipsecinit.conf` Configuration File

To enable the IPsec security policy when you start Oracle Solaris, you create a configuration file to initialize IPsec with your specific IPsec policy entries. The default name for this file is

/etc/inet/ipsecinit.conf. See the ipsecconf(8) man page for details about policy entries and their format. After the policy is configured, you can refresh the policy with the svcadm refresh ipsec/policy command.

## Sample ipsecinit.conf File

The Oracle Solaris software includes a sample IPsec policy file, ipsecinit.sample. You can use the file as a template to create your own ipsecinit.conf file. The ipsecinit.sample file contains the following examples:

```
...
# In the following simple example, outbound network traffic between the local
# host and a remote host will be encrypted. Inbound network traffic between
# these addresses is required to be encrypted as well.
#
# This example assumes that 192.0.2.11 is the IPv4 address of this host (laddr)
# and 192.0.2.12 is the IPv4 address of the remote host (raddr).
#

{laddr 192.0.2.11 raddr 192.0.2.12} ipsec
 {encr_algs aes encr_auth_algs sha256 sa shared}

# The policy syntax supports IPv4 and IPv6 addresses as well as symbolic names.
# Refer to the ipsecconf(8) man page for warnings on using symbolic names and
# many more examples, configuration options and supported algorithms.
#
# This example assumes that 192.0.2.11 is the IPv4 address of this host (laddr)
# and 192.0.2.12 is the IPv4 address of the remote host (raddr).
#
# The remote host will also need an IPsec (and IKE) configuration that mirrors
# this one.
#
# The following line will allow ssh(1) traffic to pass without IPsec protection:

{lport 22 dir both} bypass {}

#
# {laddr 192.0.2.11 dir in} drop {}
#
# Uncommenting the above line will drop all network traffic to this host unless
# it matches the rules above. Leaving this rule commented out will allow
# network packets that do not match the above rules to pass up the IP
# network stack. ...
```

### Security Considerations for `ipsecinit.conf` and `ipsecconf`

IPsec policy cannot be changed for established connections. A socket whose policy cannot be changed is called a *latched socket*. New policy entries do not protect sockets that are already latched. For more information, see the `connect`(3C) and `accept`(3C) man pages. If you are in doubt, restart the connection. For more information, see the `SECURITY` section of the `ipsecconf`(8) man page.

## `ipsecalgs` Command

The Cryptographic Framework provides authentication and encryption algorithms to IPsec. The `ipsecalgs` command can list the algorithms that each IPsec protocol supports. The `ipsecalgs` configuration is stored in the `/etc/inet/ipsecalgs` file. Typically, this file does not need to be modified and must never be edited directly. However, if you need to modify the file, use the `ipsecalgs` command. The supported algorithms are synchronized with the kernel at system boot by the `svc:/network/ipsec/ipsecalgs:default` service.

The valid IPsec protocols and algorithms are described by the ISAKMP domain of interpretation (DOI), which is covered by The Internet IP Security Domain of Interpretation for ISAKMP, RFC 2407. Specifically, the ISAKMP DOI defines the naming and numbering conventions for the valid IPsec algorithms and for their protocols, `PROTO_IPSEC_AH` and `PROTO_IPSEC_ESP`. Each algorithm is associated with exactly one protocol. These ISAKMP DOI definitions are in the `/etc/inet/ipsecalgs` file. The algorithm and protocol numbers are defined by the Internet Assigned Numbers Authority (IANA). The `ipsecalgs` command makes the list of algorithms for IPsec extensible.

For more information about the algorithms, refer to the `ipsecalgs`(8) man page. For more information about the Cryptographic Framework, see Chapter 1, "About Cryptographic Providers in Oracle Solaris" in *Managing Encryption and Certificates in Oracle Solaris 11.4*.

## `ipseckey` Command

The `ipseckey` command with various options manages keys for IPsec manually. For a description of the `ipseckey` command, see the `ipseckey`(8) man page.

### Security Considerations for `ipseckey`

The `ipseckey` command enables a role with the Network Security or Network IPsec Management rights profile to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic.

---

**Note -** Use IKE rather than manual keying, if possible.

---

For more information, see the `SECURITY` section of the `ipseckey`(8) man page.

## `kstat2` Command

The `kstat2` command can display statistics about ESP, AH, and other IPsec data. The IPsec-related options are listed in "Troubleshooting IPsec and IKE Semantic Errors" on page 205. See also the `kstat2`(8) man page.

## `snoop` Command and IPsec

The `snoop` command can parse AH and ESP headers. Because ESP encrypts its data, the `snoop` command cannot see encrypted headers that are protected by ESP. AH does not encrypt data, so traffic that is protected by AH can be inspected with the `snoop` command. The `-V` option to the command shows when AH is in use on a packet. For more details, see the `snoop`(8) man page.

For a sample of verbose `snoop` output on a protected packet, see "How to Verify That Packets Are Protected With IPsec" on page 127.

Third-party network analyzers are also available, such as the free open-source software Wireshark, which is bundled with this release.

## IPsec RFCs

The Internet Engineering Task Force (IETF) has published a number of Requests for Comment (RFCs) that describe the security architecture for the IP layer. For a link to the RFCs, see `https://www.rfc-editor.org`. The following list of RFCs covers the more general IP security references:

- RFC 2411, "IP Security Document Roadmap," November 1998

- RFC 2401, "Security Architecture for the Internet Protocol," November 1998
- RFC 2402, "IP Authentication Header," November 1998
- RFC 2406, "IP Encapsulating Security Payload (ESP)," November 1998
- RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)," November 1998
- RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP," November 1998
- RFC 2409, "The Internet Key Exchange (IKEv1)," November 1998
- RFC 5996, "Internet Key Exchange Protocol Version 2 (IKEv2)," September 2010
- RFC 3554, "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec," July 2003

## Security Associations Database for IPsec

Information on key material for IPsec security services is maintained in a security associations database (SADB). Security associations (SAs) protect inbound packets and outbound packets.

The `in.iked` daemon and the `ipseckey` command use the `PF_KEY` socket interface to maintain SADBs. For more information on how SADBs handle requests and messages, see the `pf_key`(4P) man page.

## Key Management in IPsec

The Internet Key Exchange (IKE) protocol handles key management for IPsec automatically. IPsec SAs can also be managed manually with the `ipseckey` command, but IKE is recommended. For more information, see "Key Management for IPsec Security Associations" on page 90.

The Service Management Facility (SMF) feature of Oracle Solaris provides the following key management services for IPsec:

- `svc:/network/ipsec/ike` service – The SMF service for automatic key management. The `ike` service has two instances. The `ike:ikev2` service instance runs the `in.ikev2d` daemon (IKEv2) to provide automatic key management. The `ike:default` service runs the `in.iked` daemon (IKEv1). For a description of IKE, see Chapter 8, "About Internet Key Exchange". For more information about the daemons, see the `in.ikev2d`(8) and `in.iked`(8) man pages.
- `svc:/network/ipsec/manual-key:default` service – The SMF service for manual key management. The `manual-key` service runs the `ipseckey` command with various options to

manage keys manually. For a description of the `ipseckey` command, see the `ipseckey(8)` man page.

## IKEv2 Reference

IKEv2 supersedes IKEv1. For a comparison, see "Comparison of IKEv2 and IKEv1" on page 134.

### IKEv2 Utilities and Files

The following table summarizes the configuration files for IKEv2 policy, the storage locations for IKEv2 keys, and the various commands and services that implement IKEv2. For more about services, see Chapter 1, "Introduction to the Service Management Facility" in *Managing System Services in Oracle Solaris 11.4*.

**TABLE 13** IKEv2 Service Name, Commands, and Configuration Locations

| File, Location, Command, or Service | Description | Man Page |
|---|---|---|
| `svc:/network/ipsec/ike:ikev2` | The SMF service that manages IKEv2. | `smf(7)` |
| `/usr/lib/inet/in.ikev2d` | Internet Key Exchange (IKE) daemon. Activates automated key management when the `ike:ikev2` service is enabled. | `in.ikev2d(8)` |
| `/usr/sbin/ikeadm [-v 2]` | IKE administration command for viewing and temporarily modifying the IKEv2 policy. Enables you to view IKEv2 administrative objects, such as available Diffie-Hellman groups. | `ikeadm(8)` |
| `/usr/sbin/ikev2cert` | Certificate database management command for creating and storing public key certificates as the configuration owner, `ikeuser`. Calls the `pktool` command. | `ikev2cert(8)` `pktool(1)` |
| `/etc/inet/ike/ikev2.config` | Default configuration file for the IKEv2 policy. Contains the site's rules for matching inbound IKEv2 requests and preparing outbound IKEv2 requests.<br><br>If this file exists, the `in.ikev2d` daemon starts when the `ike:ikev2` service is enabled. You can change the location of this file by using the `svccfg` command. | `ikev2.config(5)` |
| `/etc/inet/ike/ikev2.preshared` | Contains secret keys that two IKEv2 instances that are not using certificate-based authentication can use to authenticate each other. | `ikev2.preshared(5)` |
| softtoken keystore | Contains the private keys and public key certificates for IKEv2, owned by `ikeuser`. | `pkcs11_softtoken(7)` |

## IKEv2 Service

The Service Management Facility (SMF) provides the `svc:/network/ipsec/ike:ikev2` service instance to manage IKEv2. By default, this service is disabled. Before enabling this service, you must create a valid IKEv2 configuration in the `/etc/inet/ike/ikev2.config` file.

The following `ike:ikev2` service properties are configurable:

- `config_file` **property** – Specifies the location of the IKEv2 configuration file. The initial value is `/etc/inet/ike/ikev2.config`. This file has special permissions and must be owned by `ikeuser`. Do not use a different file.
- `debug_level` **property** – Sets the debugging level of the `in.ikev2d` daemon. The initial value is `op`, or operational. For possible values, see the table on debug levels under Object Types in the `ikeadm`(8) man page.
- `debug_logfile` **property** – Specifies the location of the log file for debugging IKEv2. The initial value is `/var/log/ikev2/in.ikev2d.log`.
- `fragmentation_enable` **property** – Prevents fragmentation of IKEv2 messages at the IP layer. IKEv2 fragmentation must also be enabled on the peer. The default value is `true`.
- `fragmentation_mtu` **property** – Specifies the maximum size of an IP packet carrying an IKEv2 message, in bytes, when `fragmentation_enable` is `true`. 1350 is the default. The range is from 576 to 9216.
- `kmf_policy` **property** – Sets the location of the log file for certificate policy. The default value is `/etc/inet/ike/kmf-policy.xml`. This file has special permissions and must be owned by `ikeuser`. Do not use a different file.
- `pkcs11_token/pin` **property** – Sets the PIN to use to log in to the keystore when the IKEv2 daemon starts. This value must match the value that you set for the token with the `ikev2cert setpin` command.
- `pkcs11_token/uri` **property** – Sets the PKCS #11 URI to the keystore.

For information about SMF, see Chapter 1, “Introduction to the Service Management Facility” in *Managing System Services in Oracle Solaris 11.4*. Also see the `smf`(7), `svcadm`(8), and `svccfg`(8) man pages.

## IKEv2 Daemon

The `in.ikev2d` daemon automates the management of cryptographic keys for IPsec on an Oracle Solaris system. The daemon negotiates with a remote system that is running the same protocol to provide authenticated keying materials for security associations (SAs) in a protected manner. The daemon must be running on all systems that plan to use IPsec to protect communications by using the IKEv2 protocol.

By default, the `svc:/network/ipsec/ike:ikev2` service is not enabled. After you have configured the `/etc/inet/ike/ikev2.config` file and enabled the `ike:ikev2` service instance, SMF starts the `in.ikev2d` daemon at system boot.

When the IKEv2 daemon runs, the system authenticates itself to its peer IKEv2 entity and establishes the session keys. At an interval specified in the configuration file, the IKE keys are replaced automatically. The `in.ikev2d` daemon listens for incoming IKE requests from the network and for requests for outbound traffic through the `PF_KEY` socket. For more information, see the `pf_key`(4P) man page.

Two commands support the IKEv2 daemon. The `ikeadm` command can be used to view the IKE policy. For more information, see "`ikeadm` Command for IKEv2" on page 226. The `ikev2cert` command enables you to view and manage public and private key certificates. For more information, see "IKEv2 `ikev2cert` Command" on page 226.

## IKEv2 Configuration File

The IKEv2 configuration file, `/etc/inet/ike/ikev2.config`, manages the rules that are used to negotiate the keys for the specified network endpoints that are being protected in the IPsec policy file, `/etc/inet/ipsecinit.conf`.

Key management with IKE includes rules and global parameters. An IKE rule identifies the systems or networks that the keying material secures. The rule also specifies the authentication method. Global parameters include such items as the default amount of time before an IKEv2 SA is rekeyed, `ikesa_lifetime_secs`. For examples of IKEv2 configuration files, see "Configuring IKEv2 With Preshared Keys" on page 142. For examples and descriptions of IKEv2 policy entries, see the `ikev2.config`(5) man page.

The IPsec SAs that IKEv2 supports protect the IP packets according to the policies in the IPsec configuration file, `/etc/inet/ipsecinit.conf`.

The security considerations for the `ike/ikev2.config` file are similar to the considerations for the `ipsecinit.conf` file. For details, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 220.

## `ikeadm` Command for IKEv2

When the `in.ikev2d` daemon is running, you can use the `ikeadm [-v2]` command to do the following:

- View aspects of the IKEv2 state.
- Display IKEv2 daemon objects, such as policy rules, preshared keys, available Diffie-Hellman groups, encryption and authentication algorithms, and existing active IKEv2 SAs.

For examples and a full description of this command's options, see the `ikeadm`(8) man page.

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. For details, see "Security Considerations for `ipseckey`" on page 221.

## IKEv2 Preshared Keys File

The `/etc/inet/ike/ikev2.preshared` file contains the preshared keys that are used by the IKEv2 service. The file is owned by `ikeuser` and protected at `0600`.

You must customize the default `ikev2.preshared` file when you configure a rule in the `ike/ikev2.config` file that requires preshared keys. Because IKEv2 uses these preshared keys to authenticate IKEv2 peers, this file must be valid before the `in.ikev2d` daemon reads any rules that require preshared keys.

## IKEv2 `ikev2cert` Command

The `ikev2cert` command is used to generate, store, and manage public and private keys and certificates. You use this command when the `ike/ikev2.config` file requires public key certificates. Because IKEv2 uses these certificates to authenticate IKEv2 peers, the certificates must be in place before the `in.ikev2d` daemon reads rules that require the certificates.

The `ikev2cert` command calls the `pktool` command as `ikeuser`.

The following `ikev2cert` commands manage certificates for IKEv2. The commands must be run by the `ikeuser` account. The results are stored in the PKCS #11 softtoken keystore.

- `ikev2cert setpin` – Generates a PIN for the `ikeuser` user. This PIN is required when you use certificates.

- `ikev2cert gencert` – Generates a self-signed certificate.
- `ikev2cert gencsr` – Generates a certificate signing request (CSR).
- `ikev2cert list` – Lists certificates in the keystore.
- `ikev2cert export` – Exports certificates to a file for export.
- `ikev2cert import` – Imports a certificate or CRL.

For information about the syntax of the `ikev2cert` subcommands, see the `pktool`(1) man page. For examples, see the `ikev2cert`(8) man page. For information about the softtoken keystore, see the `cryptoadm`(8) man page.

## IKEv1 Reference

The following sections provide reference information about IKEv1. IKEv1 is superseded by IKEv2, which offers faster automated key management. For more information about IKEv2, see "IKEv2 Reference" on page 223. For a comparison, see "Comparison of IKEv2 and IKEv1" on page 134.

### IKEv1 Utilities and Files

The following table summarizes the configuration files for IKEv1 policy, the storage locations for IKEv1 keys, and the various commands and services that implement IKEv1. For more about services, see Chapter 1, "Introduction to the Service Management Facility" in *Managing System Services in Oracle Solaris 11.4*.

**TABLE 14** IKEv1 Service Name, Commands, and Configuration Locations

| Service, Command, File, or Device | Description | Man Page |
|---|---|---|
| `svc:/network/ipsec/ike:default` | The SMF service that manages IKEv1. | `smf`(7) |
| `/usr/lib/inet/in.iked` | Internet Key Exchange (IKEv1) daemon. Activates automated key management when the `ike` service is enabled. | `in.iked`(8) |
| `/usr/sbin/ikeadm [-v1]` | IKE administration command for viewing and temporarily modifying the IKE policy. Enables you to view IKE administrative objects such as Phase 1 algorithms and available Diffie-Hellman groups. | `ikeadm`(8) |
| `/usr/sbin/ikecert` | Certificate database management command for manipulating local databases that hold public key certificates. | `ikecert`(8) |
| `/etc/inet/ike/config` | Default configuration file for the IKEv1 policy. Contains the site's rules for matching inbound IKEv1 requests and preparing outbound IKEv1 requests. | `ike.config`(5) |

| Service, Command, File, or Device | Description | Man Page |
|---|---|---|
| | If this file exists, the `in.iked` daemon starts when the `ike` service is enabled. You can change the location of this file by using the `svccfg` command. | |
| `ike.preshared` | Preshared keys file in the `/etc/inet/secret` directory. Contains secret keys for authentication in the Phase 1 exchange. Used when configuring IKEv1 with preshared keys. | `ike.preshared`(5) |
| `ike.privatekeys` | Private keys directory in the `/etc/inet/secret` directory. Contains the private keys that are part of a public-private key pair. | `ikecert`(8) |
| `publickeys` directory | Directory in the `/etc/inet/ike` directory that holds public keys and certificate files. Contains the public key part of a public-private key pair. | `ikecert`(8) |
| `crls` directory | Directory in the `/etc/inet/ike` directory that holds revocation lists for public keys and certificate files. | `ikecert`(8) |

## IKEv1 Service

The Service Management Facility (SMF) provides the `svc:/network/ipsec/ike:default` service to manage IKEv1. By default, this service is disabled. Before enabling this service, you must create an IKEv1 configuration file, `/etc/inet/ike/config`.

The following `ike` service properties are configurable:

- `config_file` **property –** Sets the location of the IKEv1 configuration file. The initial value is `/etc/inet/ike/config`.
- `debug_level` **property –** Sets the debugging level of the `in.iked` daemon. The initial value is `op`, or operational. For possible values, see the table on debug levels under Object Types in the `ikeadm`(8) man page.
- `admin_privilege` **property –** Sets the level of privilege of the `in.iked` daemon. The initial value is `base`. Other values are `modkeys` and `keymat`. For details, see “IKEv1 `ikeadm` Command” on page 230.

For information about SMF, see Chapter 1, “Introduction to the Service Management Facility” in *Managing System Services in Oracle Solaris 11.4*. Also see the `smf`(7), `svcadm`(8), and `svccfg`(8) man pages.

## IKEv1 Daemon

The in.iked daemon automates the management of IPsec SAs, which include the cryptographic keys that protect the packets that use IPsec. The daemon securely negotiates ISAKMP SAs and IPsec SAs with a peer system that is running the IKEv1 protocol.

By default, the svc:/network/ipsec/ike:default service is not enabled. After you have configured the /etc/inet/ike/config file and enabled the ike:default service, SMF starts the in.iked daemon at system boot. In addition to the /etc/inet/ike/config file, further configuration is stored in other files and databases, or as SMF properties. For more information, see "IKEv1 Utilities and Files" on page 227, and the ike.preshared(5), ikecert(8), and in.iked(8) man pages.

After the ike:default service is enabled, the in.iked daemon reads the configuration files and listens for external requests from an IKE peer and internal requests from IPsec for SAs.

For external requests from an IKEv1 peer, the configuration of the ike:default service determines how the daemon responds. Internal requests are routed through the PF_KEY interface. This interface handles communication between the kernel part of IPsec, which stores the IPsec SAs and performs packet encryption and decryption, and the key management daemon, in.iked, which runs in userland. When the kernel needs an SA to protect a packet, it sends a message through the PF_KEY interface to the in.iked daemon. For more information, see the pf_key(4P) man page.

Two commands support the IKEv1 daemon. The ikeadm command provides a command line interface to the running daemon. The ikecert command manages the certificate databases, ike.privatekeys and publickeys.

For more information about these commands, see the in.iked(8), ikeadm(8), and ikecert(8) man pages.

## IKEv1 Configuration File

The IKEv1 configuration file, /etc/inet/ike/config, manages the SAs for network packets that need IPsec protection according to the policies in the IPsec configuration file, /etc/inet/ipsecinit.conf.

Key management with IKE includes rules and global parameters. An IKEv1 rule identifies systems that are running another IKEv1 daemon. The rule also specifies the authentication method. For examples of IKEv1 policy files, see "Configuring IKEv2 With Preshared

Keys" on page 142. For examples and descriptions of IKEv1 policy entries, see the `ike.config(5)` man page.

The `/etc/inet/ike/config` file can include the path to a library that is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki).

The security considerations for the `ike/config` file are similar to the considerations for the `ipsecinit.conf` file. For details, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 220.

## IKEv1 `ikeadm` Command

You can use the `ikeadm` command to do the following:

- View aspects of the IKE state
- Change the properties of the IKE daemon
- Display statistics on SA creation during the Phase 1 exchange
- Debug IKE protocol exchanges
- Display IKE daemon objects, such as all Phase 1 SAs, policy rules, preshared keys, available Diffie-Hellman groups, Phase 1 encryption and authentication algorithms, and the certificate cache

For examples and a full description of this command's options, see the `ikeadm(8)` man page.

The privilege level of the running IKE daemon determines which aspects of the IKE daemon can be viewed and modified. Three levels of privilege are possible:

| | |
|---|---|
| `base` level | You cannot view or modify keys. The `base` level is the default level of privilege. |
| `keymat` level | You can view the actual keys with the `ikeadm` command. |
| `modkeys` level | You can remove, change, and add preshared keys. |

For a temporary privilege change, you can use the `ikeadm` command. For a permanent change, change the `admin_privilege` property of the `ike` service. For the temporary privilege change, see "Managing the Running IKE Daemons" on page 214.

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. See "Security Considerations for `ipseckey`" on page 221. For details that are specific to the `ikeadm` command, see the `ikeadm(8)` man page.

## IKEv1 Preshared Keys Files

When you create preshared keys manually, the keys are stored in files in the `/etc/inet/secret` directory. The `ike.preshared` file contains the preshared keys for the Phase 1 exchange when you configure a rule in the `ike/config` to use preshared keys. The `ipseckeys` file contains the preshared keys that are used to protect IP packets. The files are protected at `0600`. The `secret` directory is protected at `0700`.

Because the preshared keys are used to authenticate the Phase 1 exchange, the file must be valid before the `in.iked` daemon starts.

For examples of manually managing IPsec keys, see "How to Manually Create IPsec Keys" on page 121.

## IKEv1 Public Key Databases and Commands

The `ikecert` command manages the local system's public/private keys, public certificates, and static CRLs databases. You use this command when the IKEv1 configuration file requires public key certificates. Because IKEv1 uses these databases to authenticate the Phase 1 exchange, the databases must be populated before activating the `in.iked` daemon. Three subcommands handle each of the three databases: `certlocal`, `certdb`, and `certrldb`.

For more information, see the `ikecert`(8) man page. For information about metaslot and the softtoken keystore, see the `cryptoadm`(8) man page.

### IKEv1 `ikecert tokens` Command

The `tokens` argument lists the token IDs that are available. Token IDs enable the `ikecert certlocal` and `ikecert certdb` commands to generate public key certificates and CSRs.

### IKEv1 `ikecert certlocal` Command

The `certlocal` subcommand manages the private key database. Options to this subcommand enable you to add, view, and remove private keys. This subcommand also creates either a self-signed certificate or a CSR. The `-ks` option creates a self-signed certificate. The `-kc` option creates a CSR. Keys are stored on the system in the `/etc/inet/secret/ike.privatekeys` directory.

When you create a private key, the options to the `ikecert certlocal` command must have related entries in the `ike/config` file. The correspondences between `ikecert` options and `ike/config` entries are shown in the following table.

**TABLE 15** `ikecert` Option Correspondences to `ike/config` Entries

| `ikecert` Option | `ike/config` Entry | Description |
|---|---|---|
| `-A` *subject-alternate-name* | `cert_trust` *subject-alternate-name* | A nickname that uniquely identifies the certificate. Possible values are an IP address, an email address, or a domain name. |
| `-D` *X.509-distinguished-name* | *X.509-distinguished-name* | The full name of the certificate authority that includes the country (C), organization name (ON), organizational unit (OU), and common name (CN). |
| `-t dsa-sha1 \| dsa-sha256` | `auth_method dsa_sig` | An authentication method that is slightly slower than RSA. |
| `-t rsa-md5` and<br><br>`-t rsa-sha1 \| rsa-sha256 \| rsa-sha384 \| rsa-sha512` | `auth_method rsa_sig` | An authentication method that is slightly faster than DSA.<br><br>The RSA public key must be large enough to encrypt the biggest payload. Typically, an identity payload, such as the X.509 distinguished name, is the biggest payload. |
| `-t rsa-md5` and<br><br>`-t rsa-sha1 \| rsa-sha256 \| rsa-sha384 \| rsa-sha512` | `auth_method rsa_encrypt` | RSA encryption hides identities in IKE from eavesdroppers but requires that the IKE peers know each other's public keys. |

If you issue a CSR with the `ikecert certlocal -kc` command, you send the output of the command to a certificate authority (CA). If your company runs its own public key infrastructure (PKI), you send the output to your PKI administrator. The CA or your PKI administrator then creates certificates. The certificates that are returned to you are input to the `certdb` subcommand. The certificate revocation list (CRL) that the CA returns to you is input for the `certrldb` subcommand.

## IKEv1 `ikecert certdb` Command

The `certdb` subcommand manages the public key database. Options to this subcommand enable you to add, view, and remove certificates and public keys. The command accepts as input certificates that were generated by the `ikecert certlocal -ks` command on a remote system. For the procedure, see "How to Configure IKEv1 With Self-Signed Public Key Certificates" on page 174. This command also accepts the certificate that you receive from a CA as input. For the procedure, see "How to Configure IKEv1 With Certificates Signed by a CA" on page 179.

The certificates and public keys are stored on the system in the `/etc/inet/ike/publickeys` directory.

## IKEv1 `ikecert certrldb` Command

The `certrldb` subcommand manages the certificate revocation list (CRL) database, `/etc/inet/ike/crls`. The CRL database maintains the revocation lists for public keys. Certificates that are no longer valid are on this list. When CAs provide you with a CRL, you can install the CRL in the CRL database with the `ikecert certrldb` command. For the procedure, see "How to Handle Revoked Certificates in IKEv1" on page 185.

## IKEv1 `/etc/inet/ike/publickeys` Directory

The `/etc/inet/ike/publickeys` directory contains the public part of a public-private key pair and its certificate in files, or *slots*. The directory is protected at `0755`. The `ikecert certdb` command populates the directory.

The slots contain, in encoded form, the X.509 distinguished name of a certificate that was generated on another system. If you are using self-signed certificates, you use the certificate that you receive from the administrator of the remote system as input to the command. If you are using certificates from a CA, you install two signed certificates from the CA into this database. You install a certificate that is based on the CSR that you sent to the CA. You also install a certificate of the CA.

## IKEv1 `/etc/inet/secret/ike.privatekeys` Directory

The `/etc/inet/secret/ike.privatekeys` directory holds private key files that are part of a public-private key pair. The directory is protected at `0700`. The `ikecert certlocal` command populates the `ike.privatekeys` directory. Private keys are not effective until their public key counterparts, self-signed certificates or CAs, are installed. The public key counterparts are stored in the `/etc/inet/ike/publickeys` directory.

## IKEv1 `/etc/inet/ike/crls` Directory

The `/etc/inet/ike/crls` directory contains certificate revocation list (CRL) files. Each file corresponds to a public certificate file in the `/etc/inet/ike/publickeys` directory. CAs provide the CRLs for their certificates. You can use the `ikecert certrldb` command to populate the database.

# Network Security Glossary

These glossary entries cover words that are complex, or can be ambiguous because they are used differently in different parts of the operating system, or have different meanings in Oracle Solaris from other operating systems.

| | |
|---|---|
| **IP address** | IP addresses that are used in Oracle Solaris documentation conform to RFC5737 IPv4 Address Blocks Reserved for Documentation, RFC 5737 and IPv6 Address Prefix Reserved for Documentation, RFC 3849. |
| | IPv4 addresses used in this documentation are blocks `192.0.2.0/24`, `198.51.100.0/24`, and `203.0.113.0/24`. IPv6 addresses have prefix `2001:DB8::/32`. To show a subnet, the block is divided into multiple subnets by borrowing enough bits from the host to create the required subnet. For example, host address `192.0.2.0` might have subnets `192.0.2.32/27` and `192.0.2.64/27`. |
| **label** | 1. An IKEv2 rule's keyword whose value must match the value of the `label` keyword in a preshared key file when the value of `Sauth_method` is `preshared`. |
| | 2. A keyword used when creating an IKEv2 certificate. This value is convenient for locating all parts of the certificate (private key, public key, and public key certificate) in the keystore. |
| | 3. A mandatory access control (MAC) indication of the level of sensitivity of an object or process. Confidential and Top Secret are sample labels. Labeled network transmissions contain MAC labels. |
| | 4. An IKEv1 rule's keyword whose value is used to get the rule. |
| **subnet** | A logical subdivision of an IP network that connects systems with subnet numbers and IP address schemas, including their respective netmasks. See also IP address. |
| **trust anchor** | An alternative name for the root certificate from a certificate authority. The certificates from the root certificate to the end certificate establish a chain of trust. |
| **tunnel** | The path that is followed by a network packet while it is encapsulated. |
| | In IPsec, a configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. |

# Index