# Securing the Network in Oracle® Solaris 11.1

Part No: E28990–02
February 2013

ORACLE®

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.



130304@25097

# Contents

# Preface

This guide assumes that the Oracle Solaris operating system (Oracle Solaris OS) is installed and you are ready to secure your network.

**Note –** This Oracle Solaris release supports systems that use the SPARC and x86 families of processor architectures. The supported systems appear in the *Oracle Solaris OS: Hardware Compatibility Lists*. This document cites any implementation differences between the platform types.

## Who Should Use This Book

This book is intended for anyone responsible for administering networked systems that run Oracle Solaris. To use this book, you should have at least two years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

## Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

## Typographic Conventions

The following table describes the typographic conventions that are used in this book.

**TABLE P–1** Typographic Conventions

| Typeface | Description | Example |
| --- | --- | --- |
| `AaBbCc123` | The names of commands, files, and directories, and onscreen computer output | Edit your `.login` file.<br>Use `ls -a` to list all files.<br>`machine_name% you have mail.` |

TABLE P–1 Typographic Conventions *(Continued)*

| Typeface | Description | Example |
|---|---|---|
| **`AaBbCc123`** | What you type, contrasted with onscreen computer output | `machine_name%` **`su`**<br>`Password:` |
| *aabbcc123* | Placeholder: replace with a real name or value | The command to remove a file is `rm` *filename*. |
| *AaBbCc123* | Book titles, new terms, and terms to be emphasized | Read Chapter 6 in the *User's Guide*.<br>A *cache* is a copy that is stored locally.<br>Do *not* save the file.<br>**Note:** Some emphasized items appear bold online. |

# Shell Prompts in Command Examples

The following table shows UNIX system prompts and superuser prompts for shells that are included in the Oracle Solaris OS. In command examples, the shell prompt indicates whether the command should be executed by a regular user or a user with privileges.

TABLE P–2 Shell Prompts

| Shell | Prompt |
|---|---|
| Bash shell, Korn shell, and Bourne shell | `$` |
| Bash shell, Korn shell, and Bourne shell for superuser | `#` |
| C shell | `machine_name%` |
| C shell for superuser | `machine_name#` |

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:



## Overview of 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(1M)` 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 15.

---

# Configuring Link Protection (Task Map)

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.

| 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 13 |
| Disable link protection. | Removes link protections. | "How to Disable Link Protection" on page 14 |
| 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 14 |
| 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 15 |
| 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 15 |

## ▼ 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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

```
# dladm show-linkprop -p protection
LINK      PROPERTY      PERM VALUE    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 11 and the `dladm(1M)` 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
```

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

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

The link protection type under VALUE indicates that protection is enabled.

## ▼ 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**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     DEFAULT     POSSIBLE
vnic0      protection     rw   --        --          mac-nospoof,
                                                     restricted,
                                                     ip-nospoof,
                                                     dhcp-nospoof
```

No listing of a link protection type under VALUE indicates that link protection is disabled.

## ▼ 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 13.

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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

```
# dladm show-linkprop -p protection link
LINK     PROPERTY       PERM     VALUE          DEFAULT      POSSIBLE
link     protection     rw       ...
                                 ip-nospoof                  ip-nospoof
```

The listing of `ip-nospoof` under VALUE indicates that this protection type is enabled.

**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 `10.0.0.1` and `10.0.0.2` to the `allowed-ips` property for the `vnic0` link:

```
# dladm set-linkprop -p allowed-ips=10.0.0.1,10.0.0.2 vnic0
```

For more information, see the dladm(1M) 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 13.

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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

```
# dladm show-linkprop -p protection link
LINK     PROPERTY       PERM     VALUE          DEFAULT      POSSIBLE
link     protection     rw       ...
                                 dhcp-nospoof                dhcp-nospoof
```

The listing of `dhcp-nospoof` under `VALUE` indicates that this protection type is enabled.

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(1M) 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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           DEFAULT     POSSIBLE
vnic0    protection          rw       mac-nospoof     --          mac-nospoof,
                                      restricted                  restricted,
                                      ip-nospoof                  ip-nospoof,
                                      dhcp-nospoof                dhcp-nospoof
vnic0    allowed-ips         rw       10.0.0.1,       --          --
                                      10.0.0.2
vnic0    allowed-dhcp-cids   rw       hello           --          --
```

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

You might use the `kstat` command, but its output is not committed. For example, the following command finds the `dhcpspoofed` statistics:

```
# kstat vnic0:0:link:dhcpspoofed
module: vnic0                          instance: 0
name:   link                           class:    vnic
        dhcpspoofed                    0
```

For more information, see the `dlstat(1M)` and `kstat(1M)` man pages.

CHAPTER 2

# Tuning Your Network (Tasks)

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

## Tuning the Network (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Disable the network routing daemon. | Limits access to systems by would-be network sniffers. | "How to Disable the Network Routing Daemon" on page 18 |
| Prevent the dissemination of information about the network topology. | Prevents the broadcast of packets. | "How to Disable Broadcast Packet Forwarding" on page 19 |
| | Prevents responses to broadcast echo requests and multicast echo requests. | "How to Disable Responses to Echo Requests" on page 19 |
| 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 20 |
| 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 20 |
| 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 21 |
| Generate strong random numbers for initial TCP connections. | Complies with the sequence number generation value specified by RFC 6528. | "How to Specify a Strong Random Number for Initial TCP Connection" on page 21 |
| Prevent ICMP redirection. | Removes indicators of the network topology. | "How to Prevent ICMP Redirects" on page 22 |

| Task | Description | For Instructions |
| --- | --- | --- |
| 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 23 |

## ▼ How to Disable the Network Routing Daemon

Use this procedure to prevent network routing after installation by specifying a default router. Otherwise, perform this procedure after configuring routing manually.

---

**Note –** Many network configuration procedures require that the routing daemon be disabled. Therefore, you might have disabled this daemon as part of a larger configuration procedure.

---

**Before You Begin** You must become an administrator who is assigned the Network Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Verify that the routing daemon is running.**

```
# svcs -x svc:/network/routing/route:default
svc:/network/routing/route:default (in.routed network routing daemon)
 State: online since April 10, 2011 05:15:35 AM PDT
   See: in.routed(1M)
   See: /var/svc/log/network-routing-route:default.log
Impact: None.
```

   If the service is not running, you can stop here.

2. **Disable the routing daemon.**

```
# routeadm -d ipv4-forwarding -d ipv6-forwarding
# routeadm -d ipv4-routing -d ipv6-routing
# routeadm -u
```

3. **Verify that the routing daemon is disabled.**

```
# svcs -x routing/route:default
svc:/network/routing/route:default (in.routed network routing daemon)
 State: disabled since April 11, 2011 10:10:10 AM PDT
Reason: Disabled by an administrator.
   See: http://support.oracle.com/msg/SMF-8000-05
   See: in.routed(1M)
Impact: This service is not running.
```

**See Also** `routeadm(1M)` man page

## 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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(1M)` 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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 and _respond_to_echo_multicast (ipv4 or ipv6)" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual* and the `ipadm(1M)` 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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 (ipv4 or ipv6)" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual* and the `ipadm(1M)` man page.

For more information about the use of strict multihoming, see How to Protect a VPN With IPsec in Tunnel Mode.

## 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual* and the ipadm(1M) 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual* and the ipadm(1M) man page.

## How to Specify a Strong Random Number for Initial TCP Connection

This procedure sets the TCP initial sequence number generation parameter to comply with RFC 6528 (http://www.ietf.org/rfc/rfc6528.txt).

**Before You Begin** 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Change the default value for the `TCP_STRONG_ISS` variable.**

```
# pfedit /etc/default/inetinit
# TCP_STRONG_ISS=1
TCP_STRONG_ISS=2
```

2. **Reboot the system.**

```
# /usr/sbin/reboot
```

## ▼ 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Set the ignore redirect 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=0 ipv4
# ipadm set-prop -p _send_redirects=0 ipv6
# ipadm show-prop -p _send_redirects ipv4
PROTO PROPERTY          PERM CURRENT  PERSISTENT   DEFAULT  POSSIBLE
ipv4  _send_redirects   rw   0        0            1        0,1

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

For more information, see "_send_redirects (ipv4 or ipv6)" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual* and the `ipadm(1M)` 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

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 (ipv4 or ipv6)" in *Oracle Solaris 11.1 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" in *Oracle Solaris 11.1 Tunable Parameters Reference Manual*.

**See Also** `ipadm(1M)` man page

CHAPTER 3

# Web Servers and the Secure Sockets Layer Protocol

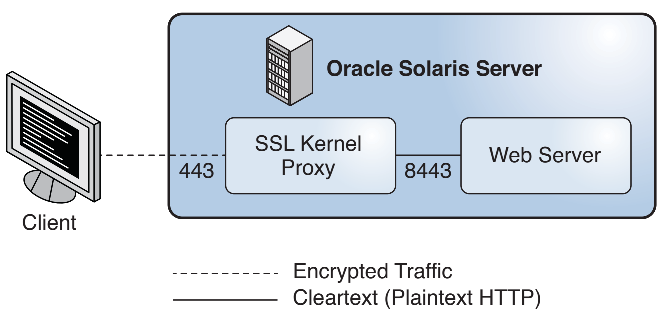
This chapter explains how to use the Secure Sockets Layer (SSL) protocol to encrypt and accelerate web server communications on your Oracle Solaris system.



## SSL Kernel Proxy Encrypts Web Server Communications

Any web server that runs on Oracle Solaris can be configured to use the SSL protocol at the kernel level, that is, the SSL kernel proxy. Examples of such web servers are the Apache 2.2 web server and the Oracle iPlanet Web Server. The SSL protocol provides confidentiality, message integrity, and endpoint authentication between two applications. When the SSL kernel proxy runs on the web server, communications are accelerated. The following illustration shows the basic configuration.

**FIGURE 3–1** Kernel-Encrypted Web Server Communications



The SSL kernel proxy implements the server side of the SSL protocol. The proxy offers several advantages.

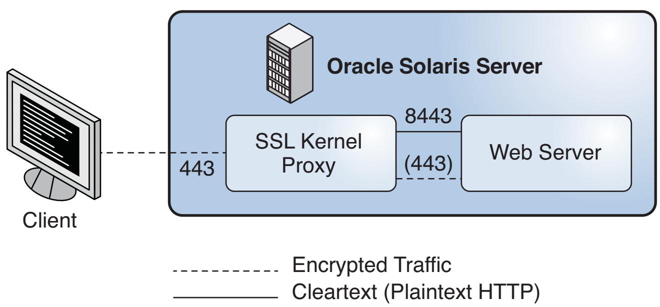
- The proxy accelerates SSL performance for server applications, like web servers, so it offers better performance than applications that rely on user-level SSL libraries. The performance improvement can be over 35 percent, depending on the workload of the application.
- The SSL kernel proxy is transparent. It has no assigned IP address. Therefore, web servers see real client IP addresses and TCP ports.
- The SSL kernel proxy and web servers are designed to work together.

  Figure 3–1 shows a basic scenario with a web server that is using the SSL kernel proxy. The SSL kernel proxy is configured on port `443`, whereas the web server is configured on port `8443`, where it receives unencrypted HTTP communications.
- The SSL kernel proxy can be configured to fall back to user-level ciphers when it does not support the requested encryption.

  Figure 3–2 shows a more complex scenario. The web server and SSL kernel proxy are configured to fall back to the user-level web server SSL.

  The SSL kernel proxy is configured on port `443`. The web server is configured on two ports. Port `8443` receives unencrypted HTTP communications, while port `443` is a fallback port. The fallback port receives encrypted SSL traffic for cipher suites that are unsupported by the SSL kernel proxy.

**FIGURE 3–2** Kernel-Encrypted Web Server Communications With User-Level Fallback Option




The SSL kernel proxy supports the SSL 3.0 and TLS 1.0 protocols, as well as most common cipher suites. See the `ksslcfg(1M)` man page for the complete list. The proxy can be configured to fall back to the user-level SSL server for any unsupported cipher suites.

# Protecting Web Servers With the SSL Kernel Proxy (Tasks)

The following procedures show how to configure web servers to use the SSL kernel proxy:


- "How to Configure an Apache 2.2 Web Server to Use the SSL Kernel Proxy" on page 27
- "How to Configure an Oracle iPlanet Web Server to Use the SSL Kernel Proxy" on page 29
- "How to Configure the SSL Kernel Proxy to Fall Back to the Apache 2.2 SSL" on page 30
- "How to Use the SSL Kernel Proxy in Zones" on page 33


## ▼ How to Configure an Apache 2.2 Web Server to Use the SSL Kernel Proxy

The SSL kernel proxy can improve the speed of SSL packet processing on an Apache 2.2 web server. This procedure implements the simple scenario that is illustrated in Figure 3–1.

**Before You Begin** You have configured an Apache 2.2 web server. This web server is included in Oracle Solaris.

You must assume the `root` role.

**1 Stop the web server.**

```
# svcadm disable svc:/network/http:apache22
```

**2 Place the server private key and the server certificate in one file.**

If only the `SSLCertificateFile` parameter is specified in the `ssl.conf` file, then the specified file can be used directly for the SSL kernel proxy.

If the `SSLCertificateKeyFile` parameter is also specified, then you must combine the certificate file and the private key file. Run a command similar to the following to combine the files:

```
# cat cert.pem key.pem > cert-and-key.pem
```

**3 Determine which parameters to use with the `ksslcfg` command.**

See the `ksslcfg(1M)` man page for the full list of options. The parameters that you *must* supply follow:

- *key-format* – Used with the `-f` option to define the certificate and key format. For the SSL kernel proxy, the supported formats are `pkcs11`, `pem`, and `pkcs12`.
- *key-and-certificate-file* – Used with the `-i` option to set the location of the file that stores the server key and the certificate for the `pem` and `pkcs12` *key-format* options.
- *password-file* – Used with the `-p` option to obtain the password used to encrypt the private key for the `pem` or `pkcs12` *key-format* options. For `pkcs11`, the password is used to authenticate to the PKCS #11 token. You must protect the password file with `0400` permissions. This file is required for unattended reboots.

- *token-label* – Used with the `-T` option to specify the PKCS #11 token.
- *certificate-label* – Used with the `-C` option to select the label in the certificate object in the PKCS #11 token.
- *proxy-port* – Used with the `-x` option to set the SSL proxy port. You must specify a different port from the standard port `80`. The web server listens on the SSL proxy port for unencrypted plaintext traffic. Typically, the value is `8443`.
- *ssl-port* – Specifies the listening port for the SSL kernel proxy. Typically, the value is `443`.

4 **Create the service instance for the SSL kernel proxy.**

Specify the SSL proxy port and associated parameters by using one of the following formats:

- **Specify PEM or PKCS #12 as the key format.**

```
# ksslcfg create -f key-format -i key-and-certificate-file \
-p password-file -x proxy-port ssl-port
```

- **Specify PKCS #11 as the key format.**

```
# ksslcfg create -f pkcs11 -T PKCS#11-token -C certificate-label \
-p password-file -x proxy-port ssl-port
```

5 **Verify that the service instance is online.**

```
# svcs svc:/network/ssl/proxy
STATE          STIME    FMRI
online         02:22:22 svc:/network/ssl/proxy:default
```

The following output indicates that the service instance was not created:

```
svcs: Pattern 'svc:/network/ssl/proxy' doesn't match any instances
STATE          STIME    FMRI
```

6 **Configure the web server to listen on the SSL proxy port.**

Edit the `/etc/apache2/2.2/http.conf` file and add a line to define the SSL proxy port. If you use the server's IP address, then the web server listens on that interface only. The line is similar to the following:

```
Listen proxy-port
```

7 **Set an SMF dependency for the web server.**

The web server service can start only after the SSL kernel proxy instance is started. The following commands establish that dependency:

```
# svccfg -s svc:/network/http:apache22
svc:/network/http:apache22> addpg kssl dependency
...apache22> setprop kssl/entities = fmri:svc:/network/ssl/proxy:kssl-INADDR_ANY-443
...apache22> setprop kssl/grouping = astring: require_all
...apache22> setprop kssl/restart_on = astring: refresh
...apache22> setprop kssl/type = astring: service
...apache22> end
```

**8 Enable the web server service.**

```
# svcadm enable svc:/network/http:apache22
```

## How to Configure an Oracle iPlanet Web Server to Use the SSL Kernel Proxy

The SSL kernel proxy can improve the speed of SSL packet processing on an Oracle iPlanet Web Server. This procedure implements the simple scenario that is illustrated in Figure 3–1.

**Before You Begin** You have installed and configured an Oracle iPlanet Web Server. The server can be downloaded from Oracle iPlanet Web Server (`http://www.oracle.com/technetwork/middleware/iplanetwebserver-098726.html?ssSourceSiteId=ocomen`). For instructions, see Oracle iPLANET WEB SERVER 7.0.15 (`http://docs.oracle.com/cd/E18958_01/index.htm`).

You must become an administrator who is assigned the Network Security rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 Stop the web server.**

Use the administrator web interface to stop the server. For instructions, see Oracle iPLANET WEB SERVER 7.0.15 (`http://docs.oracle.com/cd/E18958_01/index.htm`).

**2 Determine which parameters to use with the `ksslcfg` command.**

See the `ksslcfg`(1M) man page for the full list of options. For the list of parameters that you *must* supply, see Step 3 in "How to Configure an Apache 2.2 Web Server to Use the SSL Kernel Proxy" on page 27.

**3 Create the service instance for the SSL kernel proxy.**

Specify the SSL proxy port and associated parameters by using one of the following formats:

- **Specify PEM or PKCS #12 as the key format.**

  ```
  # ksslcfg create -f key-format -i key-and-certificate-file \
  -p password-file -x proxy-port ssl-port
  ```

- **Specify PKCS #11 as the key format.**

  ```
  # ksslcfg create -f pkcs11 -T PKCS#11-token -C certificate-label \
  -p password-file -x proxy-port ssl-port
  ```

**4 Verify that the instance is online.**

```
# svcs svc:/network/ssl/proxy
STATE          STIME    FMRI
online         02:22:22 svc:/network/ssl/proxy:default
```

**5 Configure the web server to listen on the SSL proxy port.**

For instructions, see Oracle iPLANET WEB SERVER 7.0.15 (`http://docs.oracle.com/cd/E18958_01/index.htm`).

**6 Set an SMF dependency for the web server.**

The web server service can start only after the SSL kernel proxy instance is started. The following commands establish that dependency, assuming the FMRI of the web server service is `svc:/network/http:webserver7`:

```
# svccfg -s svc:/network/http:webserver7
svc:/network/http:webserver7> addpg kssl dependency
...webserver7> setprop kssl/entities = fmri:svc:/network/ssl/proxy:kssl-INADDR_ANY-443
...webserver7> setprop kssl/grouping = astring: require_all
...webserver7> setprop kssl/restart_on = astring: refresh
...webserver7> setprop kssl/type = astring: service
...webserver7> end
```

**7 Enable the web server service.**

```
# svcadm enable svc:/network/http:webserver7
```

## ▼ How to Configure the SSL Kernel Proxy to Fall Back to the Apache 2.2 SSL

In this procedure, you configure an Apache 2.2 web server from scratch and configure the SSL kernel proxy as the primary SSL session handling mechanism. When the set of SSL ciphers that the client offers does not include a cipher that the SSL kernel proxy offers, the Apache 2.2 web server serves as a fallback mechanism w. This procedure implements the complex scenario that is illustrated in Figure 3–2.

**Before You Begin** You must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 On the Apache 2.2 web server, create a key certificate to be used by the server's SSL kernel proxy.**

**a. Generate a Certificate Signing Request (CSR).**

The following command generates a CSR and associated private key for the SSL kernel proxy:

```
# cd /root
# openssl req \
> -x509 -new \
> -subj "/C=CZ/ST=Prague region/L=Prague/CN=‘hostname‘" \
> -newkey rsa:2048 -keyout webkey.pem \
> -out webcert.pem
Generating a 2048 bit RSA private key
```

```
.+++
........+++
writing new private key to 'webkey.pem'
Enter PEM pass phrase: JohnnyCashIsCool
Verifying - Enter PEM pass phrase:  JohnnyCashIsCool
#
# chmod 440 /root/webcert.pem ; chown root:webservd /root/webcert.pem
```

For more information, see the openssl(5) man page.

**b. Send the CSR to your Certificate Authority (CA).**

**c. Replace the `webcert.pem` file with the signed certificate from your CA.**

**2 Configure the SSL kernel proxy with a passphrase and the public/private key certificate.**

**a. Create, save, and protect the passphrase.**

```
# echo "RefrigeratorsAreCool" > /root/kssl.pass
# chmod 440 /root/kssl.pass; chown root:webservd /root/kssl.pass
```

---

**Note –** The passphrase cannot contain white space.

---

**b. Combine the private key and the public key certificate into one file.**

```
# cat /root/webcert.pem /root/webkey.pem > /root/webcombo.pem
```

**c. Configure the SSL kernel proxy with the public/private key certificate and passphrase.**

```
# ksslcfg create -f pem -i /root/webcombo.pem -x 8443 -p /root/kssl.pass 443
```

**3 Configure the web server to listen on port 8443 for plaintext.**

Edit the `Listen` line in the `/etc/apache2/2.2/httpd.conf` file.

```
# pfedit /etc/apache2/2.2/httpd.conf
...
## Listen 80
Listen 8443
```

**4 Add the SSL module template, `ssl.conf`, to the Apache configuration directory.**

```
# cp /etc/apache2/2.2/samples-conf.d/ssl.conf /etc/apache2/2.2/ssl.conf
```

This module adds listening on port 443 for encrypted connections.

**5 Enable the web server to decrypt the passphrase in the `/root/kssl.pass` file.**

**a. Create a shell script that reads the `kssl.pass` file.**

```
# pfedit /root/put-passphrase.sh
#!/usr/bin/ksh -p
## Reads SSL kernel proxy passphrase
/usr/bin/cat /root/kssl.pass
```

b. **Make the script executable and protect the file.**

```
# chmod 500 /root/put-passphrase.sh
# chown webservd:webservd /root/put-passphrase.sh
```

c. **Modify the `SSLPassPhraseDialog` parameter in the `ssl.conf` file to call this shell script.**

```
# pfedit /etc/apache2/2.2/ssl.conf
...
## SSLPassPhraseDialog  builtin
SSLPassPhraseDialog exec:/root/put-passphrase.sh
```

6 **Place the web server's public and private key certificates in the correct location.**

The values of the `SSLCertificateFile` and `SSLCertificateKeyFile` parameters in the `ssl.conf` file contain the expected placement and names. You can copy or link the certificates to the correct location.

```
# ln -s  /root/webcert.pem /etc/apache2/2.2/server.crt      SSLCertificateFile default location
# ln -s /root/webkey.pem /etc/apache2/2.2/server.key     SSLCertificateKeyFile default location
```

7 **Enable the Apache service.**

```
# svcadm enable apache22
```

8 **(Optional) Verify that the two ports are working.**

Use the `openssl s_client` and `kstat` commands to view the packets.

a. **Use a cipher that is available to the SSL kernel proxy.**

```
# openssl s_client -cipher RC4-SHA -connect web-server:443
```

An increase of `1` to the `kstat` counter `kssl_full_handshakes` verifies that the SSL session was handled by the SSL kernel proxy.

```
# kstat -m kssl -s kssl_full_handshakes
```

b. **Use a cipher that is not available to the SSL kernel proxy.**

```
# openssl s_client -cipher CAMELLIA256-SHA -connect web-server:443
```

An increase of `1` to the `kstat` counter `kssl_fallback_connections` verifies that the packet arrived but the SSL session was handled by the Apache web server.

```
# kstat -m kssl -s kssl_fallback_connections
```

## Example 3–1 Configuring an Apache 2.2 Web Server to Use the SSL Kernel Proxy

The following command creates a service instance for the SSL kernel proxy that uses the `pem` key format:

```
# ksslcfg create -f pem -i cert-and-key.pem -p kssl.pass -x 8443 443
```

## ▼ How to Use the SSL Kernel Proxy in Zones

The SSL kernel proxy works in zones with the following limitations:

- All of the kernel SSL administration must be done in the global zone. The global zone administrator needs access to the local zone certificate and key files. The local zone web server can be started after the service instance is configured by using the `ksslcfg` command in the global zone.
- A specific host name or IP address must be specified with the `ksslcfg` command when you configure the instance. In particular, the instance cannot specify `INADDR_ANY` for the IP address.

**Before You Begin** The web server service is configured and enabled in the non-global zone.

You must become an administrator who is assigned the Network Security and Zone Management rights profiles. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 In the non-global zone, stop the web server.**

For example, to stop an Apache web server in the `apache-zone` zone, run the following command:

```
apache-zone # svcadm disable svc:/network/http:apache22
```

**2 In the global zone, create the service instance for the SSL kernel proxy in the zone.**

To create a service instance for the `apache-zone`, use a command similar to the following:

```
# ksslcfg create -f pem -i /zone/apache-zone/root/keypair.pem \
-p /zone/apache-zone/root/skppass -x 8443 apache-zone 443
```

**3 In the non-global zone, enable the web service instance.**

For example, enable the web service in `apache-zone`.

```
apache-zone # svcadm enable svc:/network/http:apache22
```

CHAPTER 4

# IP Filter in Oracle Solaris (Overview)

This chapter provides an overview of IP Filter, an Oracle Solaris feature. For IP Filter tasks, see Chapter 5, "IP Filter (Tasks)."

This chapter contains the following information:



## Introduction to IP Filter

The IP Filter feature of Oracle Solaris is a firewall that provides stateful packet filtering and network address translation (NAT). IP Filter also includes stateless packet filtering and the ability to create and manage address pools.

Packet filtering provides basic protection against network-based attacks. IP Filter can filter by IP address, port, protocol, network interface, and traffic direction. IP Filter can also filter by an individual source IP address, a destination IP address, by a range of IP addresses, or by address pools.

IP Filter is derived from open source IP Filter software. To view license terms, attribution, and copyright statements for open source IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If Oracle Solaris has been installed anywhere other than the default, modify the given path to access the file at the installed location.

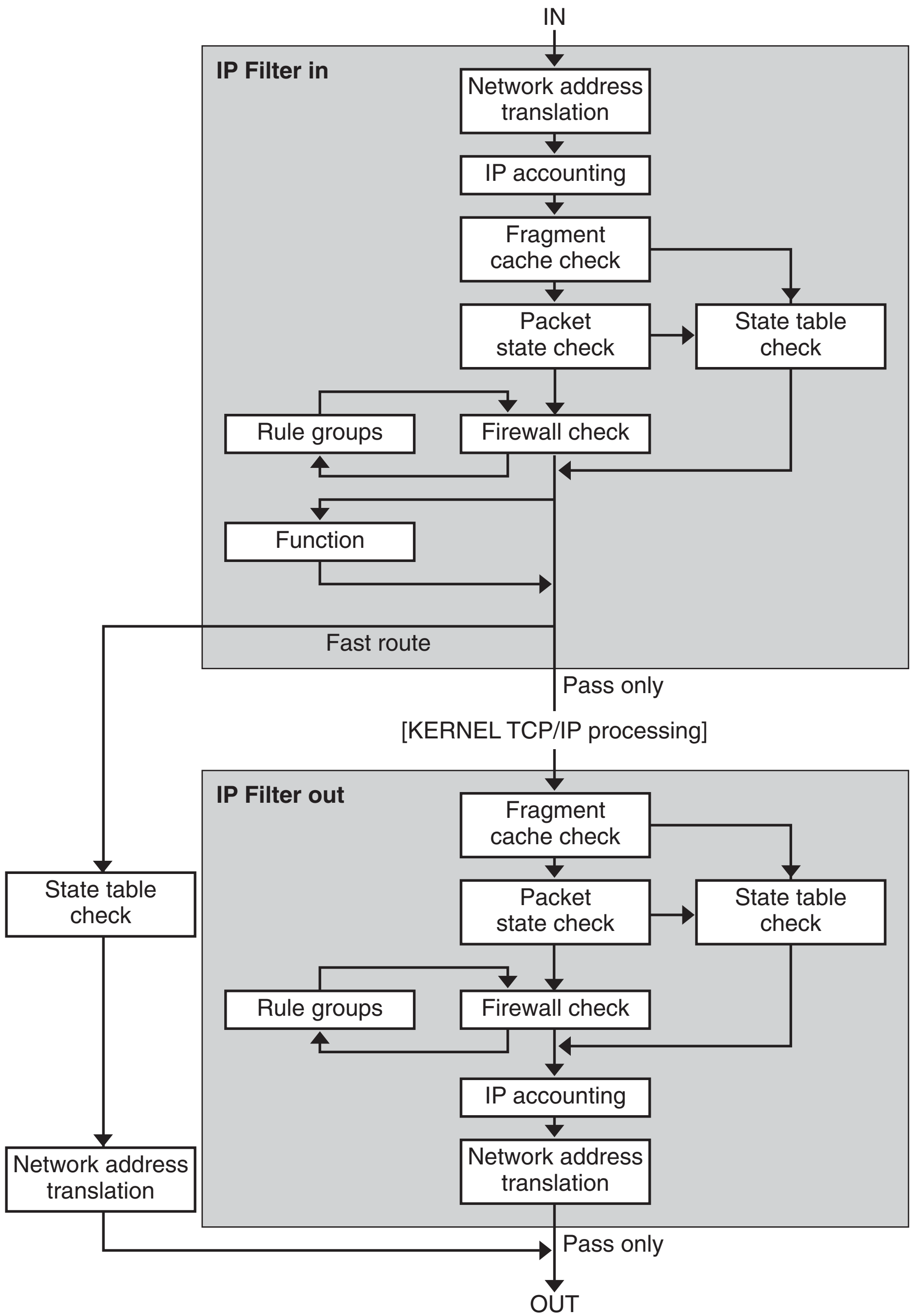
## Information Sources for Open Source IP Filter

The home page for the open source IP Filter software by Darren Reed is found at http://coombs.anu.edu.au/~avalon/ip-filter.html. This site includes information for open source IP Filter, including a link to a tutorial entitled "IP Filter Based Firewalls HOWTO" (Brendan Conoboy and Erik Fichtner, 2002). This tutorial provides step-by-step instructions for building firewalls in a BSD UNIX environment. Although written for a BSD UNIX environment, the tutorial is also relevant for the configuration of IP Filter on Oracle Solaris.

# IP Filter Packet Processing

IP Filter executes a sequence of steps as a packet is processed. The following diagram illustrates the steps of packet processing and how filtering integrates with the TCP/IP protocol stack.

**FIGURE 4–1** Packet Processing Sequence



The packet processing sequence includes the following:

- **Network Address Translation (NAT)**

  The translation of a private IP address to a different public address, or the aliasing of multiple private addresses to a single public one. NAT allows an organization to resolve the problem of IP address depletion when the organization has existing networks and needs to access the Internet.
- **IP Accounting**

  Input and output rules can be separately set up, recording the number of bytes that pass through. Each time a rule match occurs, the byte count of the packet is added to the rule and allows for collection of cascading statistics.
- **Fragment Cache Check**

  By default, fragmented packets are cached. When the all fragments for a specific packet arrive, the filtering rules are applied and either the fragments are allowed or blocked. If `set defrag off` appears in the rules file, then fragments are not cached.
- **Packet State Check**

  If `keep state` is included in a rule, all packets in a specified session are passed or blocked automatically, depending on whether the rule says `pass` or `block`.
- **Firewall Check**

  Input and output rules can be separately set up, determining whether or not a packet will be allowed through IP Filter, into the kernel's TCP/IP routines, or out onto the network.
- **Groups**

  Groups allow you to write your rule set in a tree fashion.
- **Function**

  A function is the action to be taken. Possible functions include `block`, `pass`, `literal`, and `send ICMP response`.
- **Fast-route**

  Fast-route signals IP Filter to not pass the packet into the UNIX IP stack for routing, which results in a TTL decrement.
- **IP Authentication**

  Packets that are authenticated are only passed through the firewall loops once to prevent double-processing.

# Guidelines for Using IP Filter

- IP Filter is managed by the SMF service `svc:/network/ipfilter`. For a complete overview of SMF, see Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*. For information on the step-by-step procedures that are associated with SMF, see Chapter 2, "Managing Services (Tasks)," in *Managing Services and Faults in Oracle Solaris 11.1*.
- IP Filter requires direct editing of configuration files.
- IP Filter is installed as part of Oracle Solaris. By default, the IP Filter service is enabled when your system is configured to use automatic networking. The automatic network profile, as described on the `nwam(5)` and `netadm(1M)` man pages, enables this firewall. For a custom configuration on an automatically networked system, the IP Filter service is not enabled. For the tasks associated with enabling the service, see "Configuring IP Filter" on page 47.
- To administer IP Filter, you must assume the `root` role or able to assume a role that includes the IP Filter Management rights profile. You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.
- Oracle Solaris Cluster software does not support filtering with IP Filter for scalable services, but does support IP Filter for failover services. For guidelines and restrictions when configuring IP Filter in a cluster, see "Oracle Solaris OS Feature Restrictions" in *Oracle Solaris Cluster Software Installation Guide*.
- Filtering between zones is supported provided that the IP Filter rules are implemented in a zone that functions as a virtual router for the other zones on the system.

# Using IP Filter Configuration Files

IP Filter can be used to provide firewall services or network address translation (NAT). Rules for your firewall and NAT are not provided by default. You must create custom configuration files and set the pathnames to these files as values of IP Filter service properties. After the service is enabled, these files are loaded automatically when the system is rebooted. For sample configuration files, see "IP Filter Configuration File Examples" on page 70. For more information, see the `svc.ipfd(1M)` man page.

# Using IP Filter Rule Sets

To manage your firewall, you use IP Filter to specify rule sets that you use to filter your network traffic. You can create the following types of rule sets:

- Packet filtering rule sets
- Network Address Translation (NAT) rule sets

Additionally, you can create address pools to reference groups of IP addresses. You can then use these pools later in a rule set. The address pools help to speed up rule processing. Address pools also make managing large groups of addresses easier.

# Using IP Filter's Packet Filtering Feature

You set up packet filtering by using packet filtering rule sets. Use the `ipf` command to work with packet filtering rule sets. For more information on the `ipf` command, see the `ipf`(1M) command.

You can create packet filtering rules either at the command line, using the `ipf` command, or in a packet filtering configuration file. To load the configuration file, you must create the file, then provide its pathname to the IP Filter service.

You can maintain two sets of packet filtering rule sets with IP Filter, the active rule set and the inactive rule set. In most cases, you work with the active rule set. However, the `ipf -I` command enables you to apply the command action to the inactive rule list. The inactive rule list is not used by IP Filter unless you select it. The inactive rule list provides you with a place to store rules without affecting active packet filtering.

IP Filter processes the rules in the rules list from the beginning of the configured rules list to the end of the rules list before passing or blocking a packet. IP Filter maintains a flag that determines whether it will or will not pass a packet. It goes through the entire rule set and determines whether to pass or block the packet based on the last matching rule.

There are two exceptions to this process. The first exception is if the packet matches a rule containing the `quick` keyword. If a rule includes the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked. The second exception is if the packet matches a rule containing the `group` keyword. If a packet matches a group, only rules tagged with the group are checked.

## Configuring Packet Filtering Rules

Use the following syntax to create packet filtering rules:

*action* `[in|out]` *option keyword, keyword...*

1. Each rule begins with an action. IP Filter applies the action to the packet if the packet matches the rule. The following list includes the commonly used actions applied to a packet.

| | |
|---|---|
| `block` | Prevents the packet from passing through the filter. |
| `pass` | Allows the packet through the filter. |
| `log` | Logs the packet but does not determine if the packet is blocked or passed. Use the `ipmon` command to view the log. |

| | |
|---|---|
| `count` | Includes the packet in the filter statistics. Use the `ipfstat` command to view the statistics. |
| `skip` *number* | Makes the filter skip over *number* filtering rules. |
| `auth` | Requests that packet authentication be performed by a user program that validates packet information. The program determines whether the packet is passed or blocked. |

2. Following the action, the next word must be either `in` or `out`. Your choice determines whether the packet filtering rule is applied to an incoming packet or to an outgoing packet.
3. Next, you can choose from a list of options. If you use more than one option, they must be in the order shown here.

| | |
|---|---|
| `log` | Logs the packet if the rule is the last matching rule. Use the `ipmon` command to view the log. |
| `quick` | Executes the rule containing the `quick` option if there is a packet match. All further rule checking stops. |
| `on` *interface-name* | Applies the rule only if the packet is moving in or out of the specified interface. |
| `dup-to` *interface-name* | Copies the packet and sends the duplicate out on *interface-name* to an optionally specified IP address. |
| `to` *interface-name* | Moves the packet to an outbound queue on *interface-name*. |

4. After specifying the options, you can choose from a variety of keywords that determine whether the packet matches the rule. The following keywords must be used in the order shown here.

---

**Note –** By default, any packet that does not match any rule in the configuration file is passed through the filter.

---

| | |
|---|---|
| `tos` | Filters 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. The time-to-live value stored in a packet indicates the length of time a packet can be on the network before being discarded. |
| `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` can be used to match either a TCP or a UDP packet. |

| | |
|---|---|
| `from/to/all/any` | Matches any or all of the following: the source IP address, the destination IP address, and the port number. The `all` keyword is used to accept packets from all sources and to all destinations. |
| `with` | Matches specified attributes associated with the packet. Insert either the word `not` or the word `no` in front of the keyword in order to match the packet only if the option is not present. |
| `flags` | Used for TCP to filter based on TCP flags that are set. For more information on the TCP flags, see the `ipf(4)` man page. |
| `icmp-type` | Filters according to 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. |
| `keep` *keep-options* | Determines the information that is kept for a packet. The *keep-options* that are available include the `state` option. The `state` option keeps information about the session and can be kept for TCP, UDP, and ICMP packets. |
| `head` *number* | Creates a new group for filtering rules, which is denoted by the number *number*. |
| `group` *number* | Adds the rule to group number *number* instead of the default group. All filtering rules are placed in group 0 if no other group is specified. |

The following example illustrates how to put together the packet filtering rule syntax to create a rule. To block incoming traffic from the IP address `192.168.0.0/16`, you would include the following rule in the rule list:

```
block in quick from 192.168.0.0/16 to any
```

For the complete grammar and syntax used to write packet filtering rules, see the `ipf(4)` man page. For tasks associated with packet filtering, see "Managing Packet Filtering Rule Sets for IP Filter" on page 53. For an explanation of the IP address scheme (`192.168.0.0/16`) shown in the example, see Chapter 1, "Planning the Network Deployment," in *Configuring and Administering Oracle Solaris 11.1 Networks*.

# Using IP Filter's NAT Feature

NAT sets up mapping rules that translate source and destination IP addresses into other Internet or intranet addresses. These rules modify the source and destination addresses of incoming or outgoing IP packets and send the packets on. You can also use NAT to redirect traffic from one port to another port. NAT maintains the integrity of the packet during any modification or redirection done on the packet.

You can create NAT rules either at the command line, using the `ipnat` command, or in a NAT configuration file. You must create the NAT configuration file and set its pathname as the value of the `config/ipnat_config_file` property of the service. The default value is `/etc/ipf/ipnat.conf`. For more information, see the `ipnat(1M)` command.

NAT rules can apply to both IPv4 and IPv6 addresses. However, you cannot specify both types of addresses in a single rule. Instead, you must set separate rules for each address type. In a NAT rule that includes IPv6 addresses, you cannot use the `mapproxy` and `rdrproxy` NAT commands simultaneously.

## Configuring NAT Rules

Use the following syntax to create NAT rules:

*command interface-name parameters*

1. Each rule begins with one of the following commands:

   `map` — Maps one IP address or network to another IP address or network in an unregulated round-robin process.

   `rdr` — Redirects packets from one IP address and port pair to another IP address and port pair.

   `bimap` — Establishes a bidirectional NAT between an external IP address and an internal IP address.

   `map-block` — Establishes static IP address-based translation. This command is based on an algorithm that forces addresses to be translated into a destination range.
2. Following the command, the next word is the interface name, such as `bge0`.
3. Next, you can choose from a variety of parameters, which determine the NAT configuration. Some of the parameters include:

   `ipmask` — Designates the network mask.

   `dstipmask` — Designates the address that `ipmask` is translated to.

   `mapport` — Designates `tcp`, `udp`, or `tcp/udp` protocols, along with a range of port numbers.

The following example illustrates how to construct a NAT rule. To rewrite a packet that goes out on the `net2` device with a source address of `192.168.1.0/24` and to externally show its source address as `10.1.0.0/16`, you would include the following rule in the NAT rule set:

```
map net2 192.168.1.0/24 -> 10.1.0.0/16
```

The following rules apply to IPv6 addresses:

```
map net3 fec0:1::/64 -> 2000:1:2::/72 portmap tcp/udp 1025:65000
map-block net3 fe80:0:0:209::/64 -> 209:1:2::/72 ports auto
rdr net0 209::ffff:fe13:e43e port 80 -> fec0:1::e,fec0:1::f port 80 tcp round-robin
```

For the complete grammar and syntax, see the `ipnat(4)` man page.

# Using IP Filter's Address Pools Feature

Address pools establish a single reference that is used to name a group of address/netmask pairs. Address pools provide processes to reduce the time needed to match IP addresses with rules. Address pools also make managing large groups of addresses easier.

Address pool configuration rules can reside in a file that is loaded by the IP Filter service. You must create a file, and set its pathname as the value of the `config/ippool_config_file` property of the service. The default value is `/etc/ipf/ippool.conf`.

## Configuring Address Pools

Use the following syntax to create an address pool:

`table role =` *role-name* `type =` *storage-format* `number =` *reference-number*

`table` Defines the reference for the multiple addresses.

`role` Specifies the role of the pool in IP Filter. At this time, the only role you can reference is `ipf`.

`type` Specifies the storage format for the pool.

`number` Specifies the reference number that is used by the filtering rule.

For example, to reference the group of addresses `10.1.1.1` and `10.1.1.2`, and the network `192.16.1.0` as pool number `13`, you would include the following rule in the address pool configuration file:

```
table role = ipf type = tree number = 13
{ 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24 };
```

Then, to reference pool number `13` in a filtering rule, you would construct the rule similar to the following example:

```
pass in from pool/13 to any
```

Note that you must load the pool file before loading the rules file that contains a reference to the pool. If you do not, the pool is undefined, as shown in the following output:

```
# ipfstat -io
empty list for ipfilter(out)
block in from pool/13(!) to any
```

Even if you add the pool later, the addition of the pool does not update the kernel rule set. You also need to reload the rules file that references the pool.

For the complete grammar and syntax, see the ippool(4) man page.

# IPv6 for IP Filter

IPv6 packet filtering can filter based on the source/destination IPv6 address, pools containing IPv6 addresses, and IPv6 extension headers.

IPv6 is similar to IPv4 in many ways. However, header and packet size differ between the two versions of IP, which is an important consideration for IP Filter. IPv6 packets known as *jumbograms* contain a datagram longer than 65,535 bytes. IP Filter does not support IPv6 jumbograms. To learn more about other IPv6 features, see "Major Features of IPv6" in *System Administration Guide: IP Services*.

---

**Note** – For more information on jumbograms, refer to the document IPv6 Jumbograms, RFC 2675 from the Internet Engineering Task Force (IETF). [http://www.ietf.org/rfc/rfc2675.txt]

---

IP Filter tasks associated with IPv6 do not differ substantially from IPv4. The most notable difference is the use of the `-6` option with certain commands. Both the `ipf` command and the `ipfstat` command include the `-6` option for use with IPv6 packet filtering. Use the `-6` option with the `ipf` command to load and flush IPv6 packet filtering rules. To display IPv6 statistics, use the `-6` option with the `ipfstat` command. The `ipmon` and `ippool` commands also support IPv6, although there is no associated option for IPv6 support. The `ipmon` command has been enhanced to accommodate the logging of IPv6 packets. The `ippool` command supports the pools with IPv6 addresses. You can create separate pools for IPv4 and IPv6 addresses, or a pool containing both IPv4 and IPv6 addresses.

To create re-usable IPv6 packet filtering rules, you must create a specific IPv6 file. Then, you set its pathname as the value of the `config/ip6_config_file` property of the IP Filter service. The default value is `/etc/ipf/ip6.conf`.

For more information on IPv6, see Chapter 3, "Introducing IPv6 (Overview)," in *System Administration Guide: IP Services*. For tasks associated with IP Filter, see Chapter 5, "IP Filter (Tasks)."

# IP Filter Man Pages

The following table describes the man page documentation relevant to IP Filter.

| Man Page | Description |
| --- | --- |
| ipf(1M) | Manages IP Filter rules, displays tunables, and performs other tasks. |
| ipf(4) | Contains the grammar and syntax for creating IP Filter packet filtering rules. |
| ipfilter(5) | Describes IP Filter software. |
| ipfs(1M) | Saves and restores NAT information and state table information across reboots. |
| ipfstat(1M) | Retrieves and displays statistics on packet processing. |
| ipmon(1M) | Opens the log device and views logged packets for both packet filtering and NAT. |
| ipnat(1M) | Manages NAT rules and displays NAT statistics. |
| ipnat(4) | Contains the grammar and syntax for creating NAT rules. |
| ippool(1M) | Creates and manages address pools. |
| ippool(4) | Contains the grammar and syntax for creating IP Filter address pools. |
| svc.ipfd(1M) | Provides information about configuring the IP Filter service. |

CHAPTER 5

# IP Filter (Tasks)

This chapter provides step-by-step instructions for tasks. For overview information about IP Filter, see Chapter 4, "IP Filter in Oracle Solaris (Overview)."

This chapter contains the following information:



## Configuring IP Filter

The following task map identifies the procedures to create IP Filter rules, and enable and disable the service.

**TABLE 5–1** Configuring IP Filter (Task Map)

| Task | For Instructions |
| --- | --- |
| View the files that IP Filter uses and the status of the service. | "How to Display IP Filter Service Defaults" on page 48 |
| Customize packet filtering rule sets for network traffic, packets over a NAT, and address pools. | "How to Create IP Filter Configuration Files" on page 48 |
| Enable, refresh, or disable the IP Filter service. | "How to Enable and Refresh IP Filter" on page 50 |
| Modify the default setting for packets that arrive in fragments. | "How to Disable Packet Reassembly" on page 50 |
| Filter traffic between zones on your system. | "How to Enable Loopback Filtering" on page 51 |
| Stop using IP Filter.. | "How to Disable Packet Filtering" on page 52 |

## How to Display IP Filter Service Defaults

**Before You Begin** To run the ipfstat command, you must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **View the configuration file names and locations for the IP Filter service.**

```
% svccfg -s ipfilter:default listprop   | grep file
config/ipf6_config_file                       astring     /etc/ipf/ipf6.conf
config/ipnat_config_file                      astring     /etc/ipf/ipnat.conf
config/ippool_config_file                     astring     /etc/ipf/ippool.conf
firewall_config_default/custom_policy_file    astring     none
```

The first three file properties have suggested file locations. These files do not exist until you create them. You can change the location of a configuration file by changing the property value for that file. For the procedure, see "How to Create IP Filter Configuration Files" on page 48.

You modify the fourth file property when you customize your own packet filtering rules. See Step 1 and Step 2 in "How to Create IP Filter Configuration Files" on page 48.

2. **Determine if the IP Filter service is enabled.**

- On a manually networked system, IP Filter is not enabled by default.

```
% svcs -x ipfilter:default
svc:/network/ipfilter:default (IP Filter)
 State: disabled since Mon Sep 10 10:10:50 2012
Reason: Disabled by an administrator.
   See: http://oracle.com/msg/SMF-8000-05
   See: ipfilter(5)
Impact: This service is not running.
```

- On an automatically networked system on an IPv4 network, run the following command to view the IP Filter policy:

```
$ ipfstat -io
```

To view the file that created the policy, read /etc/nwam/loc/NoNet/ipf.conf. This file is for viewing only. To modify the policy, see "How to Create IP Filter Configuration Files" on page 48.

---

**Note –** To view IP Filter policy on an IPv6 network, add the -6 option, as in: ipfstat -6io. For more information, see the ipfstat(1M) man page.

---

## How to Create IP Filter Configuration Files

To modify the IP Filter policy for an automatically configured network configuration or to use IP Filter in a manually configured network, you create configuration files, inform the service about these files, then enable the service.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Specify the file location of the policy file for the IP Filter service.**

   This file contains the packet filtering rule set.

   a. **First. you set the policy file to `custom`.**

```
$ svccfg -s ipfilter:default setprop firewall_config_default/policy = astring: "custom"
```

   b. **Then, you specify the location.**

   For example, make `/etc/ipf/myorg.ipf.conf` the location of your packet filtering rule set.

```
$ svccfg -s ipfilter:default \
setprop firewall_config_default/custom_policy_file = astring: "/etc/ipf/myorg.ipf.conf"
```

2. **Create your packet filtering rule set.**

   For information about packet filtering, see "Using IP Filter's Packet Filtering Feature" on page 40. For examples of configuration files, see "IP Filter Configuration File Examples" on page 70, and the `/etc/nwam/loc/NoNet/ipf.conf` file.

---

**Note –** If your specified policy file is empty, no filtering occurs. An empty packet filtering file is the same as having a rule set that reads:

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

---

3. **(Optional) Create a network address translation (NAT) configuration file for IP Filter.**

   To filter packets over a NAT, create a file for your NAT rules with an appropriate name, such as `/etc/ipf/ipnat.conf`. To change this name, change the value of the `config/ipnat_config_file` service property, as in:

```
$ svccfg -s ipfilter:default \
setprop config/ipnat_config_file = astring: "/etc/ipf/myorg.ipnat.conf"
```

   For more information about NAT, see "Using IP Filter's NAT Feature" on page 42.

4. **(Optional) Create an address pool configuration file.**

   To refer to a group of addresses as a single address pool, create a file for the pool with an appropriate name, such as `/etc/ipf/ippool.conf`. To change this name, change the value of the `config/ippool_config_file` service property, as in:

```
$ svccfg -s ipfilter:default \
setprop config/ippool_config_file = astring: "/etc/ipf/myorg.ippool.conf"
```

   An address pool can contain any combination of IPv4 and IPv6 addresses. For more information about address pools, see "Using IP Filter's Address Pools Feature" on page 44.

5 **(Optional) Enable filtering of loopback traffic.**

If you intend to filter traffic between zones that are configured in your system, you must enable loopback filtering. See "How to Enable Loopback Filtering" on page 51. You must also define rule sets that apply to the zones.

6 **(Optional) Disable the reassembly of fragmented packets.**

By default, fragments are reassembled in IP Filter. To modify the default, see "How to Disable Packet Reassembly" on page 50

## How to Enable and Refresh IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

You have completed "How to Create IP Filter Configuration Files" on page 48.

1 **Enable IP Filter.**

To enable IP Filter initially, type the following command:

```
$ svcadm enable network/ipfilter
```

2 **After you modify IP Filter configuration files when the service is running, refresh the service.**

```
$ svcadm refresh network/ipfilter
```

---

**Note –** The `refresh` command briefly disables the firewall. To retain the firewall, append rules or add a new configuration file. For procedures with examples, see "Working With IP Filter Rule Sets" on page 53.

---

## How to Disable Packet Reassembly

By default, fragments are reassembled in IP Filter. To disable this reassembly, you insert a rule at the beginning of your policy file.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile and the `solaris.admin.edit/`*path-to-IPFilter-policy-file* authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1 **Disable IP Filter.**

```
$ svcadm disable network/ipfilter
```

**2 Add the following rule at the beginning of your IP Filter policy file.**

```
set defrag off;
```

Use the `pfedit` command, as in:

```
$ pfedit /etc/ipf/myorg.ipf.conf
```

This rule must precede all `block` and `pass` rules that are defined in the file. However, you can insert comments before the line, similar to the following example:

```
# Disable fragment reassembly
#
set defrag off;
# Define policy
#
block in all
block out all
other rules
```

**3 Enable IP Filter.**

```
$ svcadm enable network/ipfilter
```

**4 Verify that packets are not being reassembled.**

```
$ ipf -T defrag
defrag  min 0   max 0x1 current 0
```

If `current` is `0`, fragments are not being reassembled. If `current` is `1`, fragments are being reassembled.

## ▼ How to Enable Loopback Filtering

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile and the `solaris.admin.edit/`*path-to-IPFilter-policy-file* authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 Stop IP Filter if it is running.**

```
$ svcadm disable network/ipfilter
```

**2 Add the following rule at the beginning of your IP Filter policy file.**

```
set intercept_loopback true;
```

Use the `pfedit` command, as in:

```
$ pfedit /etc/ipf/myorg.ipf.conf
```

This line must precede all `block` and `pass` rules that are defined in the file. However, you can insert comments before the line, similar to the following example:

```
...
#set defrag off;
#
# Enable loopback filtering to filter between zones
#
set intercept_loopback true;
#
# Define policy
#
block in all
block out all
other rules
```

3 **Enable IP Filter.**

```
$ svcadm enable network/ipfilter
```

4 **To verify the status of loopback filtering, use the following command:**

```
$ ipf -T ipf_loopback
ipf_loopback    min 0   max 0x1 current 1
$
```

If `current` is `0`, loopback filtering is disabled. If `current` is `1`, loopback filtering is enabled.

## ▼ How to Disable Packet Filtering

This procedure removes all rules from the kernel and disables the service. If you use this procedure, you must enable IP Filter with the appropriate configuration files to restart packet filtering and NAT. For more information, see "How to Enable and Refresh IP Filter" on page 50.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

● **To disable the service, use the `svcadm` command.**

```
$ svcadm disable network/ipfilter
```

To test or debug the service, you can remove rule sets while the service is running. For more information, see "Working With IP Filter Rule Sets" on page 53.

# Working With IP Filter Rule Sets

You might want to modify or deactivate packet filtering and NAT rules under the following circumstances:

- For testing purposes
- To troubleshoot system problems when you think the problems are caused by IP Filter

The following task map identifies the procedures associated with IP Filter rule sets.

**TABLE 5–2** Working With IP Filter Rule Sets (Task Map)

| Task | For Instructions |
| --- | --- |
| View the active packet filtering rule set. | "How to View the Active Packet Filtering Rule Set" on page 54 |
| View an inactive packet filtering rule set. | "How to View the Inactive Packet Filtering Rule Set" on page 54 |
| Activate a different active rule set. | "How to Activate a Different or Updated Packet Filtering Rule Set" on page 54 |
| Remove a rule set. | "How to Remove a Packet Filtering Rule Set" on page 55 |
| Add rules to the rule sets. | "How to Append Rules to the Active Packet Filtering Rule Set" on page 56<br>"How to Append Rules to the Inactive Packet Filtering Rule Set" on page 57 |
| Move between active and inactive rule sets. | "How to Switch Between Active and Inactive Packet Filtering Rule Sets" on page 58 |
| Delete an inactive rule set from the kernel. | "How to Remove an Inactive Packet Filtering Rule Set From the Kernel" on page 59 |
| View active NAT rules. | "How to View Active NAT Rules in IP Filter" on page 59 |
| Remove NAT rules. | "How to Deactivate NAT Rules in IP Filter" on page 60 |
| Add rules to active NAT rules. | "How to Append Rules to the NAT Packet Filtering Rules" on page 60 |
| View active address pools. | "How to View Active Address Pools" on page 61 |
| Remove an address pool. | "How to Remove an Address Pool" on page 62 |
| Add rules to an address pool. | "How to Append Rules to an Address Pool" on page 62 |

# Managing Packet Filtering Rule Sets for IP Filter

IP Filter allows both active and inactive packet filtering rule sets to reside in the kernel. The active rule set determines what filtering is being done on incoming packets and outgoing packets. The inactive rule set also stores rules. These rules are not used unless you make the inactive rule set the active rule set. You can manage, view, and modify both active and inactive packet filtering rule sets.

---

**Note –** The following procedures provide examples for IPv4 networks. For IPv6 packets, use the `-6` option, as described in Step 2 of "How to Display IP Filter Service Defaults" on page 48.

---

## How to View the Active Packet Filtering Rule Set

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the active packet filtering rule set.**

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

```
$ ipfstat -io
empty list for ipfilter(out)
pass in quick on net1 from 192.168.1.0/24 to any
pass in all
block in on net1 from 192.168.1.10/32 to any
```

## How to View the Inactive Packet Filtering Rule Set

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the inactive packet filtering rule set.**

  The following example shows output from the inactive packet filtering rule set.

```
$ ipfstat -I -io
pass out quick on net1 all
pass in quick on net1 all
```

## How to Activate a Different or Updated Packet Filtering Rule Set

Use the following procedure if you want to perform either of the following tasks:

- Activate a packet filtering rule set other than the one that is currently in use by IP Filter.
- Reload the same filtering rule set that has been newly updated.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Choose one of the following steps:**
   - Create a new rule set in a separate file if you want to activate an entirely different rule set.
   - Update the current rule set in your configuration file.

2 **Remove the current rule set and load the new rule set.**

$ **ipf -Fa -f** *filename*

The rules in *filename* replace the active rule set.

---

**Note –** Do not use commands such as `ipf -D` or `svcadm restart` to load the updated rule set. Such commands expose your network because they disable the firewall before loading the new rule set.

---

**Example 5–1** Activating a Different Packet Filtering Rule Set

The following example shows how to replace one packet filtering rule set with a different rule set.

```
$ ipfstat -io
empty list for ipfilter(out)
pass in quick on net0 all
$ ipf -Fa -f /etc/ipf/ipfnew.conf
$ ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
```

**Example 5–2** Reloading an Updated Packet Filtering Rule Set

The following example shows how to reload a packet filtering rule set that is currently active and which is then updated.

```
$ ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any

(Edit the /etc/ipf/myorg.ipf.conf configuration file.)

$ svcadm refresh network/ipfilter
$ ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any
block in quick on net11 from 192.168.0.0/12 to any
```

## ▼ How to Remove a Packet Filtering Rule Set

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

● **Remove the rule set.**

$ **ipf -F [a|i|o]**

`-a` Removes all filtering rules from the rule set.

`-i` Removes the filtering rules for incoming packets.

`-o` Removes the filtering rules for outgoing packets.

**Example 5–3** Removing a Packet Filtering Rule Set

The following example shows how to remove all filtering rules from the active filtering rule set.

```
$ ipfstat -io
block out log on net0 all
block in log quick from 10.0.0.0/8 to any
$ ipf -Fa
$ ipfstat -io
empty list for ipfilter(out)
empty list for ipfilter(in)
```

## ▼ How to Append Rules to the Active Packet Filtering Rule Set

Appending rules to an existing rule set can be useful when testing or debugging. The IP Filter service remains enabled when the rules are added. However, when the service is refreshed, restarted, or enabled, the rules are lost, unless they exist in files that are a property of the IP Filter service.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

● **Use one of the following methods to append rules to the active rule set:**

- Append rules to the rule set at the command line using the `ipf -f -` command.

  ```
  $ echo "block in on net1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
  ```

  These appended rules are not part of IP Filter configuration when the service is refreshed, restarted, or enabled.

- Perform the following commands:

  a. Create a rule set in a file of your choice.

  b. Add the rules that you have created to the active rule set.

  ```
  $ ipf -f filename
  ```

  The rules in *filename* are added to the end of the active rule set. Because IP Filter uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

If *filename* is the value of one of the IP Filter configuration file properties, then the rules are reloaded when the service is enabled, restarted, or refreshed. Otherwise, the appended rules provide a temporary rule set.

**Example 5–4** Appending Rules to the Active Packet Filtering Rule Set

The following example shows how to add a rule to the active packet filtering rule set from the command line.

```
$ ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
$ echo "block in on net1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
$ ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on net1 proto tcp from 10.1.1.1/32 to any
```

## ▼ How to Append Rules to the Inactive Packet Filtering Rule Set

Creating an inactive rule set in the kernel can be useful when testing or debugging. The rule set can be switched with the active rule set without stopping the IP Filter service. However, when the service is refreshed, restarted, or enabled, the inactive rule set must be added.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Create a rule set in a file of your choice.**

2. **Add the rules that you have created to the inactive rule set.**

   `$ ipf -I -f` *filename*

   The rules in *filename* are added to the end of the inactive rule set. Because IP Filter uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

**Example 5–5** Appending Rules to the Inactive Rule Set

The following example shows how to add a rule to the inactive rule set from a file.

```
$ ipfstat -I -io
pass out quick on net1 all
pass in quick on net1 all
$ ipf -I -f /etc/ipf/ipftrial.conf
```

```
$ ipfstat -I -io
pass out quick on net1 all
pass in quick on net1 all
block in log quick from 10.0.0.0/8 to any
```

## How to Switch Between Active and Inactive Packet Filtering Rule Sets

Switching to a different rule set in the kernel can be useful when testing or debugging. The rule set can be made active without stopping the IP Filter service.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Switch the active and inactive rule sets.**

  ```
  $ ipf -s
  ```

  This command enables you to switch between the active and inactive rule sets in the kernel. Note that if the inactive rule set is empty, there is no packet filtering.

---

**Note –** When the IP Filter service is refreshed, restarted, or enabled, the rules that are in files that are properties of the IP Filter service are restored. The inactive rule set is not restored.

---

### Example 5–6 Switching Between the Active and Inactive Packet Filtering Rule Sets

The following example shows how using the `ipf -s` command results in the inactive rule set becoming the active rule set and the active rule set becoming the inactive rule set.

- Before running the `ipf -s` command, the output from the `ipfstat -I -io` command shows the rules in the inactive rule set. The output from the `ipfstat -io` command shows the rules in the active rule set.

  ```
  $ ipfstat -io
  empty list for ipfilter(out)
  block in log quick from 10.0.0.0/8 to any
  block in on net1 proto tcp from 10.1.1.1/32 to any
  $ ipfstat -I -io
  pass out quick on net1 all
  pass in quick on net1 all
  block in log quick from 10.0.0.0/8 to any
  ```

- After running the `ipf -s` command, the output from the `ipfstat -I -io` and the `ipfstat -io` command show that the content of the two rules sets have switched.

  ```
  $ ipf -s
  Set 1 now inactive
  $ ipfstat -io
  pass out quick on net1 all
  pass in quick on net1 all
  ```

```
block in log quick from 10.0.0.0/8 to any
$ ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on net1 proto tcp from 10.1.1.1/32 to any
```

## How to Remove an Inactive Packet Filtering Rule Set From the Kernel

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Specify the inactive rule set in the "flush all" command.**

```
$ ipf -I -Fa
```

---

**Note –** If you subsequently run `ipf -s`, the empty inactive rule set will become the active rule set. An empty active rule set means that *no* filtering will be done.

---

**Example 5–7** Removing an Inactive Packet Filtering Rule Set From the Kernel

The following example shows how to flush the inactive packet filtering rule set so that all rules have been removed.

```
$ ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on net1 proto tcp from 10.1.1.1/32 to any
$ ipf -I -Fa
$ ipfstat -I -io
empty list for inactive ipfilter(out)
empty list for inactive ipfilter(in)
```

# Managing NAT Rules for IP Filter

Use the following procedures to manage, view, and modify NAT rules for IP Filter.

## How to View Active NAT Rules in IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the active NAT rules.**

  The following example shows the output from the active NAT rules set.

```
$ ipnat -l
List of active MAP/Redirect filters:
map net0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

## How to Deactivate NAT Rules in IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Remove NAT rules from the kernel.**

```
$ ipnat -FC
```

  The `-C` option removes all entries in the current NAT rule listing. The `-F` option removes all active entries in the current NAT translation table, which shows the currently active NAT mappings.

**Example 5–8** Removing NAT Rules

The following example shows how to remove the entries in the current NAT rules.

```
$ ipnat -l
List of active MAP/Redirect filters:
map net0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
$ ipnat -C
1 entries flushed from NAT list
$ ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
```

## How to Append Rules to the NAT Packet Filtering Rules

Appending rules to an existing rule set can be useful when testing or debugging. The IP Filter service remains enabled when the rules are added. However, when the service is refreshed, restarted, or enabled, the NAT rules are lost, unless they exist in a file that is a property of the IP Filter service.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Use one of the following methods to append rules to the active rule set:**
  - Append rules to the NAT rule set at the command line using the `ipnat -f -` command.

    ```
    $ echo "map net0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
    ```

    These appended rules are not part of IP Filter configuration when the service is refreshed, restarted, or enabled.
  - Perform the following commands:

    a. Create additional NAT rules in a file of your choice.

    b. Add the rules that you have created to the active NAT rules.

    ```
    $ ipnat -f filename
    ```

    The rules in *filename* are added to the end of the NAT rules.

    If *filename* is the value of one of the IP Filter configuration file properties, then the rules are reloaded when the service is enabled, restarted, or refreshed. Otherwise, the appended rules provide a temporary rule set.

**Example 5–9** Appending Rules to the NAT Rule Set

The following example shows how to add a rule to the NAT rule set from the command line.

```
$ ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
$ echo "map net0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
$ ipnat -l
List of active MAP/Redirect filters:
map net0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

# Managing Address Pools for IP Filter

Use the following procedures to manage, view, and modify address pools.

## ▼ How to View Active Address Pools

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the active address pool.**

  The following example shows how to view the contents of the active address pool.

```
$ ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

## How to Remove an Address Pool

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Remove the entries in the current address pool.**

```
$ ippool -F
```

**Example 5–10** Removing an Address Pool

The following example shows how to remove an address pool.

```
$ ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
$ ippool -F
1 object flushed
$ ippool -l
```

## How to Append Rules to an Address Pool

Appending rules to an existing rule set can be useful when testing or debugging. The IP Filter service remains enabled when the rules are added. However, when the service is refreshed, restarted, or enabled, the address pool rules are lost, unless they exist in a file that is a property of the IP Filter service.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Use one of the following methods to append rules to the active rule set:**

   - Append rules to the rule set at the command line using the `ippool -f -` command.

     ```
     $ echo "table role = ipf type = tree number = 13
     {10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24};" | ippool -f -
     ```

     These appended rules are not part of IP Filter configuration when the service is refreshed, restarted, or enabled.

- Perform the following commands:
  a. Create additional address pools in a file of your choice.
  b. Add the rules that you have created to the active address pool.

     $ **ippool -f** *filename*

     The rules in *filename* are added to the end of the active address pool.

2 **If the rules contain pools that are not in the original rule set, perform the following steps:**

  a. **Add the pools to a new packet filtering rule.**

  b. **Append the new packet filtering rule to the current rule set.**

     Follow the instructions in "How to Append Rules to the Active Packet Filtering Rule Set" on page 56.

---

**Note –** Do not refresh or restart the IP Filter service, You will lose your added address pool rules.

---

**Example 5–11** Appending Rules to an Address Pool

The following example shows how to add an address pool to the address pool rule set from the command line.

```
$ ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
$ echo "table role = ipf type = tree number = 100
 {10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24};" | ippool -f -
$ ippool -l
table role = ipf type = tree number = 100
        { 10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24; };
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

# Displaying Statistics and Information for IP Filter

**TABLE 5–3** Displaying IP Filter Statistics and Information (Task Map)

| Task | For Instructions |
| --- | --- |
| View state tables. | "How to View State Tables for IP Filter" on page 64 |
| View statistics about packet state. | "How to View State Statistics for IP Filter" on page 65 |

**TABLE 5–3** Displaying IP Filter Statistics and Information (Task Map) *(Continued)*

| Task | For Instructions |
| --- | --- |
| List IP Filter tunables. | "How to View IP Filter Tunable Parameters" on page 65 |
| View NAT statistics. | "How to View NAT Statistics for IP Filter" on page 66 |
| View address pool statistics. | "How to View Address Pool Statistics for IP Filter" on page 66 |

## ▼ How to View State Tables for IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

● **View the state table.**

```
$ ipfstat
```

---

**Note –** You can use the `-t` option to view the state table in the UNIX `top` utility format.

---

**Example 5–12** Viewing State Tables for IP Filter

The following example shows state table output.

```
$ ipfstat
bad packets:            in 0    out 0
 IPv6 packets:          in 56286 out 63298
 input packets:         blocked 160 passed 11 nomatch 1 counted 0 short 0
output packets:         blocked 0 passed 13681 nomatch 6844 counted 0 short 0
 input packets logged:  blocked 0 passed 0
output packets logged:  blocked 0 passed 0
 packets logged:        input 0 output 0
 log failures:          input 0 output 0
fragment state(in):     kept 0  lost 0  not fragmented 0
fragment reassembly(in):bad v6 hdr 0     bad v6 ehdr 0  failed reassembly 0
fragment state(out):    kept 0  lost 0  not fragmented 0
packet state(in):       kept 0  lost 0
packet state(out):      kept 0  lost 0
ICMP replies:   0       TCP RSTs sent:  0
Invalid source(in):     0
Result cache hits(in):  152     (out):  6837
IN Pullups succeeded:   0       failed: 0
OUT Pullups succeeded:  0       failed: 0
Fastroute successes:    0       failures:       0
TCP cksum fails(in):    0       (out):  0
IPF Ticks:      14341469
Packet log flags set: (0)
        none
```

## ▼ How to View State Statistics for IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the state statistics.**

```
$ ipfstat -s
```

**Example 5–13** Viewing State Statistics for IP Filter

The following example shows state statistics output.

```
$ ipfstat -s
IP states added:
        0 TCP
        0 UDP
        0 ICMP
        0 hits
        0 misses
        0 maximum
        0 no memory
        0 max bucket
        0 active
        0 expired
        0 closed
State logging enabled

State table bucket statistics:
        0 in use
        0.00% bucket usage
        0 minimal length
        0 maximal length
        0.000 average length
```

## ▼ How to View IP Filter Tunable Parameters

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the kernel tunable parameters for IP Filter.**

  The following output is truncated.

```
$ ipf -T list
fr_flags    min 0    max 0xffffffff    current 0
fr_active    min 0    max 0    current 0
...
ipstate_logging    min 0    max 0x1    current 1
...
```

```
fr_authq_ttl    min 0x1    max 0x7fffffff    current sz = 0
fr_enable_rcache    min 0    max 0x1    current 0
```

## How to View NAT Statistics for IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View NAT statistics.**

```
$ ipnat -s
```

**Example 5–14** Viewing NAT Statistics for IP Filter

The following example shows NAT statistics.

```
$ ipnat -s
mapped  in      0       out     0
added   0       expired 0
no memory       0       bad nat 0
inuse   0
rules   1
wilds   0
```

## How to View Address Pool Statistics for IP Filter

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View address pool statistics.**

```
$ ippool -s
```

**Example 5–15** Viewing Address Pool Statistics for IP Filter

The following example shows address pool statistics.

```
$ ippool -s
Pools:  3
Hash Tables:    0
Nodes:  0
```

# Working With Log Files for IP Filter

TABLE 5–4 Working With IP Filter Log Files (Task Map)

| Task | For Instructions |
|---|---|
| Create a separate IP Filter log file. | "How to Set Up a Log File for IP Filter" on page 67 |
| View state, NAT, and normal log files. | "How to View IP Filter Log Files" on page 68 |
| Flush the packet log buffer. | "How to Flush the Packet Log Buffer" on page 69 |
| Save logged packets to a file for later reference. | "How to Save Logged Packets to a File" on page 70 |

## ▼ How to Set Up a Log File for IP Filter

By default, all log information for IP Filter is recorded in the `syslogd` file. It is good practice to create a log file to record IP Filter traffic information separately from other data that might be logged in the default log file.

**Before You Begin** You must assume the `root` role.

1. **Determine which `system-log` service instance is online.**

```
# svcs system-log
STATE          STIME    FMRI
disabled       13:11:55 svc:/system/system-log:rsyslog
online         13:13:27 svc:/system/system-log:default
```

---

**Note –** If the `rsyslog` service instance is online, modify the `rsyslog.conf` file.

---

2. **Edit the `/etc/syslog.conf` file by adding the following two lines:**

```
# Save IP Filter log output to its own file
local0.debug             /var/log/log-name
```

---

**Note –** In your entry, use the Tab key, not the Spacebar, to separate `local0.debug` from `/var/log/`*log-name*. For more information, see the `syslog.conf(4)` and `syslogd(1M)` man pages.

---

3. **Create the new log file.**

```
# touch /var/log/log-name
```

4. **Refresh the configuration information for the `system-log` service.**

```
# svcadm refresh system-log:default
```

**Note** – Refresh the `system-log:rsyslog` service instance if the `rsyslog` service is online.

**Example 5–16** Creating an IP Filter Log

The following example shows how to create `ipmon.log` to archive IP Filter information.

In `/etc/syslog.conf`:

```
## Save IP Filter log output to its own file
local0.debug<Tab>/var/log/ipmon.log
```

At the command line:

```
# touch /var/log/ipmon.log
# svcadm restart system-log
```

## How to View IP Filter Log Files

**Before You Begin** You have completed "How to Set Up a Log File for IP Filter" on page 67.

You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View the state, NAT, or normal log files.**

  To view a log file, type the following command, using the appropriate option:

  `# ipmon -o [S|N|I]` *filename*

  `S` Displays the state log file.

  `N` Displays the NAT log file.

  `I` Displays the normal IP log file.

  - **To view all state, NAT, and normal log files, use all the options:**

    `# ipmon -o SNI` *filename*

  - **After you stop the `ipmon` daemon, you can use the `ipmon` command to display state, NAT, and IP filter log files:**

    ```
    # pkill ipmon
    # ipmon -a filename
    ```

**Note –** Do not use the `ipmon -a` syntax if the `ipmon` daemon is still running. Normally, the daemon is automatically started during system boot. Issuing the `ipmon -a` command also opens another copy of `ipmon`. In such a case, both copies read the same log information, and only one gets a particular log message.

For more information about viewing log files, see the `ipmon(1M)` man page.

**Example 5–17** Viewing IP Filter Log Files

The following example shows the output from `/var/ipmon.log`.

```
# ipmon -o SNI /var/ipmon.log
02/09/2012 15:27:20.606626 net0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

or

```
# pkill ipmon
# ipmon -aD /var/ipmon.log
02/09/2012 15:27:20.606626 net0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

## ▼ How to Flush the Packet Log Buffer

This procedure clears the buffer and displays the output on the screen.

**Before You Begin** You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Flush the packet log buffer.**

```
# ipmon -F
```

**Example 5–18** Flushing the Packet Log Buffer

The following example shows the output when a log file is removed. The system provides a report even when there is nothing stored in the log file, as in this example.

```
# ipmon -F
0 bytes flushed from log buffer
0 bytes flushed from log buffer
0 bytes flushed from log buffer
```

## ▼ How to Save Logged Packets to a File

You can save packets to a file during debugging, or when you want to audit the traffic manually.

**Before You Begin** You must assume the root role.

● **Save the logged packets to a file.**

```
# cat /dev/ipl > filename
```

Continue logging packets to the *filename* file until you interrupt the procedure by typing `Control-C` to get the command line prompt back.

**Example 5–19** Saving Logged Packets to a File

The following example shows the result when logged packets are saved to a file.

```
# cat /dev/ipl > /tmp/logfile
^C#

# ipmon -f /tmp/logfile
02/09/2012 15:30:28.708294 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 52 -S IN
02/09/2012 15:30:28.708708 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2012 15:30:28.792611 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 70 -AP IN
02/09/2012 15:30:28.872000 net0 @0:1 p 129.146.157.149,33923 ->
 129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2012 15:30:28.872142 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 43 -AP IN
02/09/2012 15:30:28.872808 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2012 15:30:28.872951 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 47 -AP IN
02/09/2012 15:30:28.926792 net0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
.
.
(output truncated)
```

# IP Filter Configuration File Examples

The following examples illustrate packet filtering rules that apply to a single host, a server, and a router.

Configuration files follow standard UNIX syntax rules:

- The pound sign (#) indicates a line containing comments.
- Rules and comments can coexist on the same line.
- Extraneous white space is allowed to keep rules easy to read.
- Rules can be more than one line long. Use the backslash (\) at the end of a line to indicate that the rule continues on the next line.

For more detailed syntax information, see "Configuring Packet Filtering Rules" on page 40.

**EXAMPLE 5–20** IP Filter Host Configuration

This example shows a configuration on a host machine with a `net0` network interface.

```
# pass and log everything by default
pass in log on net0 all
pass out log on net0 all

# block, but don't log, incoming packets from other reserved addresses
block in quick on net0 from 10.0.0.0/8 to any
block in quick on net0 from 172.16.0.0/12 to any

# block and log untrusted internal IPs. 0/32 is notation that replaces
# address of the machine running IP Filter.
block in log quick from 192.168.1.15 to <thishost>
block in log quick from 192.168.1.43 to <thishost>

# block and log X11 (port 6000) and remote procedure call
# and portmapper (port 111) attempts
block in log quick on net0 proto tcp from any to net0/32 port = 6000 keep state
block in log quick on net0 proto tcp/udp from any to net0/32 port = 111 keep state
```

This rule set begins with two unrestricted rules that allow everything to pass into and out of the `net0` interface. The second set of rules blocks any incoming packets from the private address spaces `10.0.0.0` and `172.16.0.0` from entering the firewall. The next set of rules blocks specific internal addresses from the host machine. Finally, the last set of rules blocks packets coming in on port 6000 and port 111.

**EXAMPLE 5–21** IP Filter Server Configuration

This example shows a configuration for a host machine acting as a web server. This machine has an `net0` network interface.

```
# web server with an net0 interface
# block and log everything by default;
# then allow specific services
# group 100 - inbound rules
# group 200 - outbound rules
# (0/32) resolves to our IP address)
*** FTP proxy ***
```

**EXAMPLE 5–21** IP Filter Server Configuration *(Continued)*

```
# block short packets which are packets
# fragmented too short to be real.
block in log quick all with short

# block and log inbound and outbound by default,
# group by destination
block in log on net0 from any to any head 100
block out log on net0 from any to any head 200


# web rules that get hit most often
pass in quick on net0 proto tcp from any \
to net0/32 port = http flags S keep state group 100
pass in quick on net0 proto tcp from any \
to net0/32 port = https flags S keep state group 100


# inbound traffic - ssh, auth
pass in quick on net0 proto tcp from any \
to net0/32 port = 22 flags S keep state group 100
pass in log quick on net0 proto tcp from any \
to net0/32 port = 113 flags S keep state group 100
pass in log quick on net0 proto tcp from any port = 113 \
to net0/32 flags S keep state group 100

# outbound traffic - DNS, auth, NTP, ssh, WWW, smtp
pass out quick on net0 proto tcp/udp from net0/32 \
to any port = domain flags S keep state group 200
pass in quick on net0 proto udp from any \
port = domain to net0/32 group 100

pass out quick on net0 proto tcp from net0/32 \
to any port = 113 flags S keep state group 200
pass out quick on net0 proto tcp from net0/32 port = 113 \
to any flags S keep state group 200

pass out quick on net0 proto udp from net0/32 to any \
port = ntp group 200
pass in quick on net0 proto udp from any \
port = ntp to net0/32 port = ntp group 100

pass out quick on net0 proto tcp from net0/32 \
to any port = ssh flags S keep state group 200

pass out quick on net0 proto tcp from net0/32 \
to any port = http flags S keep state group 200
pass out quick on net0 proto tcp from net0/32 \
to any port = https flags S keep state group 200

pass out quick on net0 proto tcp from net0/32 \
to any port = smtp flags S keep state group 200


# pass icmp packets in and out
pass in quick on net0 proto icmp from any to net0/32  keep state group 100
```

**EXAMPLE 5–21** IP Filter Server Configuration *(Continued)*

```
pass out quick on net0 proto icmp from net0/32 to any keep state group 200

# block and ignore NETBIOS packets
block in quick on net0 proto tcp from any \
to any port = 135 flags S keep state group 100

block in quick on net0 proto tcp from any port = 137 \
to any flags S keep state group 100
block in quick on net0 proto udp from any to any port = 137 group 100
block in quick on net0 proto udp from any port = 137 to any group 100

block in quick on net0 proto tcp from any port = 138 \
to any flags S keep state group 100
block in quick on net0 proto udp from any port = 138 to any group 100

block in quick on net0 proto tcp from any port = 139 to any flags S keep state
group 100
block in quick on net0 proto udp from any port = 139 to any group 100
```

**EXAMPLE 5–22** IP Filter Router Configuration

This example shows a configuration for a router that has an internal interface, `net0`, and an external interface, `net1`.

```
# internal interface is net0 at 192.168.1.1
# external interface is net1 IP obtained via DHCP
# block all packets and allow specific services
*** NAT ***
*** POOLS ***

# Short packets which are fragmented too short to be real.
block in log quick all with short

# By default, block and log everything.
block in log on net0 all
block in log on net1 all
block out log on net0 all
block out log on net1 all


# Packets going in/out of network interfaces that aren't on the loopback
# interface should not exist.
block in log quick on net0 from 127.0.0.0/8 to any
block in log quick on net0 from any to 127.0.0.0/8
block in log quick on net1 from 127.0.0.0/8 to any
block in log quick on net1 from any to 127.0.0.0/8


# Deny reserved addresses.
block in quick on net1 from 10.0.0.0/8 to any
block in quick on net1 from 172.16.0.0/12 to any
block in log quick on net1 from 192.168.1.0/24 to any
block in quick on net1 from 192.168.0.0/16 to any
```

**EXAMPLE 5–22** IP Filter Router Configuration *(Continued)*

```
# Allow internal traffic
pass in quick on net0 from 192.168.1.0/24 to 192.168.1.0/24
pass out quick on net0 from 192.168.1.0/24 to 192.168.1.0/24


# Allow outgoing DNS requests from our servers on .1, .2, and .3
pass out quick on net1 proto tcp/udp from net1/32 to any port = domain keep state
pass in quick on net0 proto tcp/udp from 192.168.1.2 to any port = domain keep state
pass in quick on net0 proto tcp/udp from 192.168.1.3 to any port = domain keep state


# Allow NTP from any internal hosts to any external NTP server.
pass in quick on net0 proto udp from 192.168.1.0/24 to any port = 123 keep state
pass out quick on net1 proto udp from any to any port = 123 keep state


# Allow incoming mail
pass in quick on net1 proto tcp from any to net1/32 port = smtp keep state
pass in quick on net1 proto tcp from any to net1/32 port = smtp keep state
pass out quick on net1 proto tcp from 192.168.1.0/24 to any port = smtp keep state


# Allow outgoing connections: SSH, WWW, NNTP, mail, whois
pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = 22 keep state
pass out quick on net1 proto tcp from 192.168.1.0/24 to any port = 22 keep state

pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass out quick on net1 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = 443 keep state
pass out quick on net1 proto tcp from 192.168.1.0/24 to any port = 443 keep state

pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = nntp keep state
block in quick on net1 proto tcp from any to any port = nntp keep state
pass out quick on net1 proto tcp from 192.168.1.0/24 to any port = nntp keep state

pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = smtp keep state

pass in quick on net0 proto tcp from 192.168.1.0/24 to any port = whois keep state
pass out quick on net1 proto tcp from any to any port = whois keep state


# Allow ssh from offsite
pass in quick on net1 proto tcp from any to net1/32 port = 22 keep state


# Allow ping out
pass in quick on net0 proto icmp all keep state
pass out quick on net1 proto icmp all keep state


# allow auth out
pass out quick on net1 proto tcp from net1/32 to any port = 113 keep state
pass out quick on net1 proto tcp from net1/32 port = 113 to any keep state
```

**EXAMPLE 5–22** IP Filter Router Configuration *(Continued)*

```
# return rst for incoming auth
block return-rst in quick on net1 proto tcp from any to any port = 113 flags S/SA


# log and return reset for any TCP packets with S/SA
block return-rst in log on net1 proto tcp from any to any flags S/SA

# return ICMP error packets for invalid UDP packets
block return-icmp(net-unr) in proto udp all
```

CHAPTER 6

# IP Security Architecture (Overview)

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

This chapter contains the following information:



To implement IPsec on your network, see Chapter 7, "Configuring IPsec (Tasks)." For reference information, see Chapter 8, "IP Security Architecture (Reference)."

## Introduction to IPsec

IPsec protects IP packets by authenticating the packets, by encrypting the packets, or by doing both. IPsec is performed inside the IP module. Therefore, an Internet 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 protection involves the following main components:

- **Security protocols –** The IP datagram protection mechanisms. The authentication header (AH) includes a hash of the IP packet and ensures integrity. The content of the datagram is not encrypted, but the receiver is assured that the packet contents have not been altered. The receiver is also assured that the packets were sent by the sender. The encapsulating security payload (ESP) encrypts IP data, thus obscuring the content during packet transmission. ESP also can ensure data integrity through an authentication algorithm option.
- **Security associations (SA) –** The cryptographic parameters and the IP security protocol as applied to a specific flow of network traffic. Each SA has a unique reference called the Security Parameters Index (SPI).
- **Security associations database (SADB) –** The database that associates a security protocol with an IP destination address and an indexing number. The indexing number is called the security parameter index (SPI). These three elements (the security protocol, the destination address, and the SPI) uniquely identify a legitimate IPsec packet. The database ensures that a protected packet that arrives to the packet destination is recognized by the receiver. The receiver also uses information from the database to decrypt the communication, verify that the packets are unchanged, reassemble the packets, and deliver the packets to their ultimate destination.
- **Key management –** The generation and distribution of keys for the cryptographic algorithms and for the SPI.
- **Security mechanisms –** The authentication and encryption algorithms that protect the data in the IP datagrams.
- **Security policy database (SPD) –** The database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine how the packets should be processed. A packet can be discarded. A packet can be passed in the clear. Or, a packet can be protected with IPsec. For outbound packets, the SPD and the SADB determine what level of protection to apply. For inbound packets, the SPD helps to determine if the level of protection on the packet is acceptable. If the packet is protected by IPsec, the SPD is consulted after the packet has been decrypted and has been verified.

IPsec applies the security mechanisms to IP datagrams that travel to the IP destination address. The receiver uses information in its SADB to verify that the arriving packets are legitimate and to decrypt them. Applications can invoke IPsec to apply security mechanisms to IP datagrams 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 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. All RFCs are copyrighted by the Internet Society. For a link to the RFCs, see `http://www.ietf.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 (IKE)," November 1998
- RFC 3554, "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec," July 2003

## IPsec Terminology

The IPsec RFCs define a number of terms that are useful to recognize when implementing IPsec on your systems. The following table lists IPsec terms, provides their commonly used acronyms, and defines each term. For a list of terminology used in key negotiation, see Table 9–1.

**TABLE 6–1** IPsec Terms, Acronyms, and Uses

| IPsec Term | Acronym | Definition |
|---|---|---|
| Security association | SA | The cryptographic parameters and the IP security protocol that are applied to a specific flow of network traffic. The SA is defined by a triplet: a security protocol, a unique security parameter index (SPI), and an IP destination. |
| Security associations database | SADB | Database that contains all active security associations. |
| Security parameter index | SPI | The indexing value for a security association. An SPI is a 32-bit value that distinguishes among SAs that have the same IP destination and security protocol. |
| Security policy database | SPD | Database that determines if outbound packets and inbound packets have the specified level of protection. |

**TABLE 6–1** IPsec Terms, Acronyms, and Uses *(Continued)*

| IPsec Term | Acronym | Definition |
| --- | --- | --- |
| Key exchange | | The process of generating keys by using asymmetric cryptographic algorithms. The two main methods are RSA and Diffie-Hellman. |
| Diffie-Hellman | DH | A key exchange algorithm that allows key generation and key authentication. Often called *authenticated key exchange*. |
| RSA | RSA | A key exchange algorithm that allows key generation and key distribution. The protocol is named for its three creators, Rivest, Shamir, and Adleman. |
| Internet Security Association and Key Management Protocol | ISAKMP | The common framework for establishing the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP is the IETF standard for handling an IKE exchange. |

# IPsec Packet Flow

Figure 6–1 shows how an IP addressed packet, as part of an IP datagram, 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. How to apply these entities, as well as how to choose the algorithms, are described in subsequent sections.

Figure 6–2 shows the IPsec inbound process.

**FIGURE 6–1** IPsec Applied to Outbound Packet Process

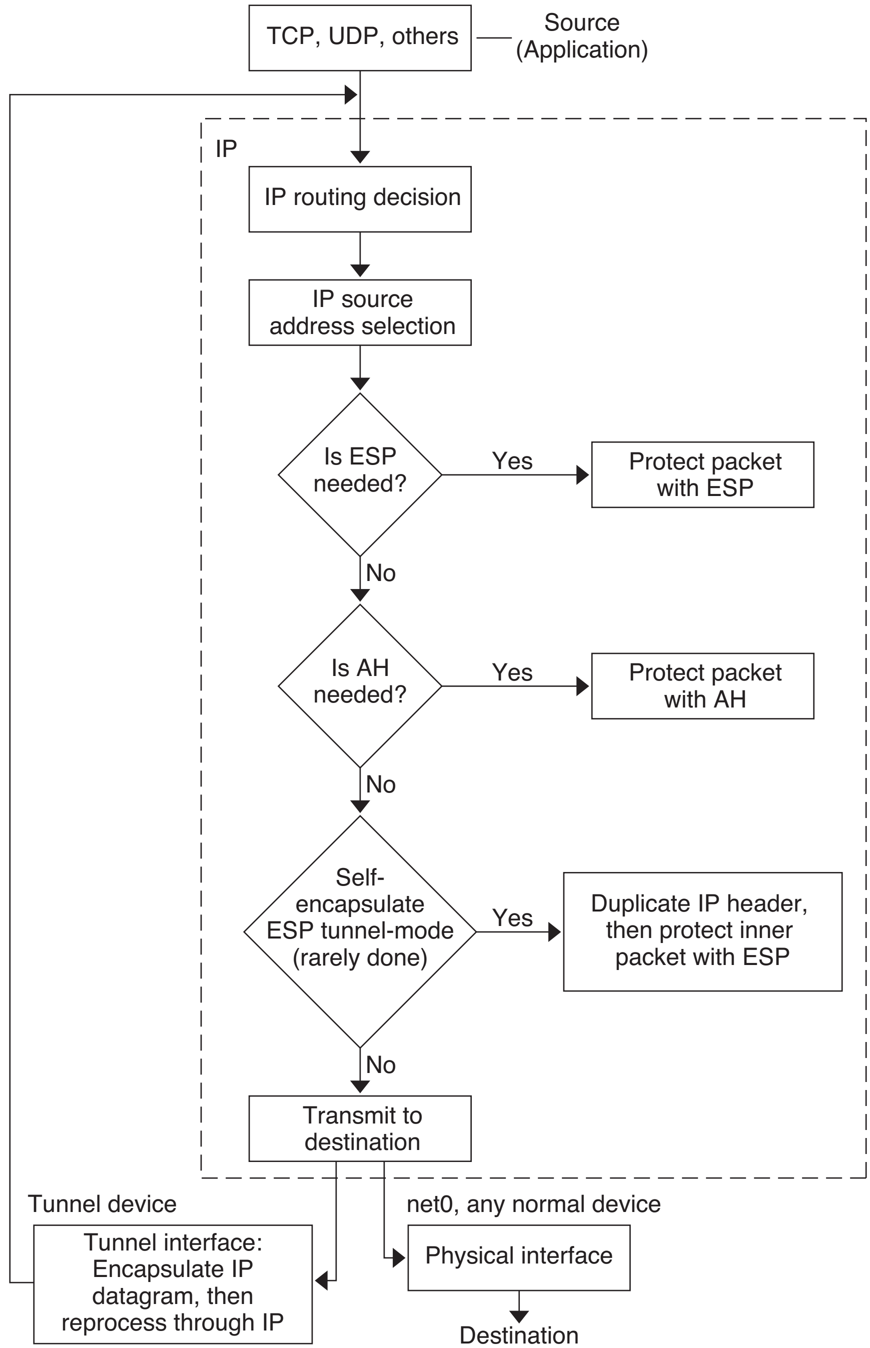
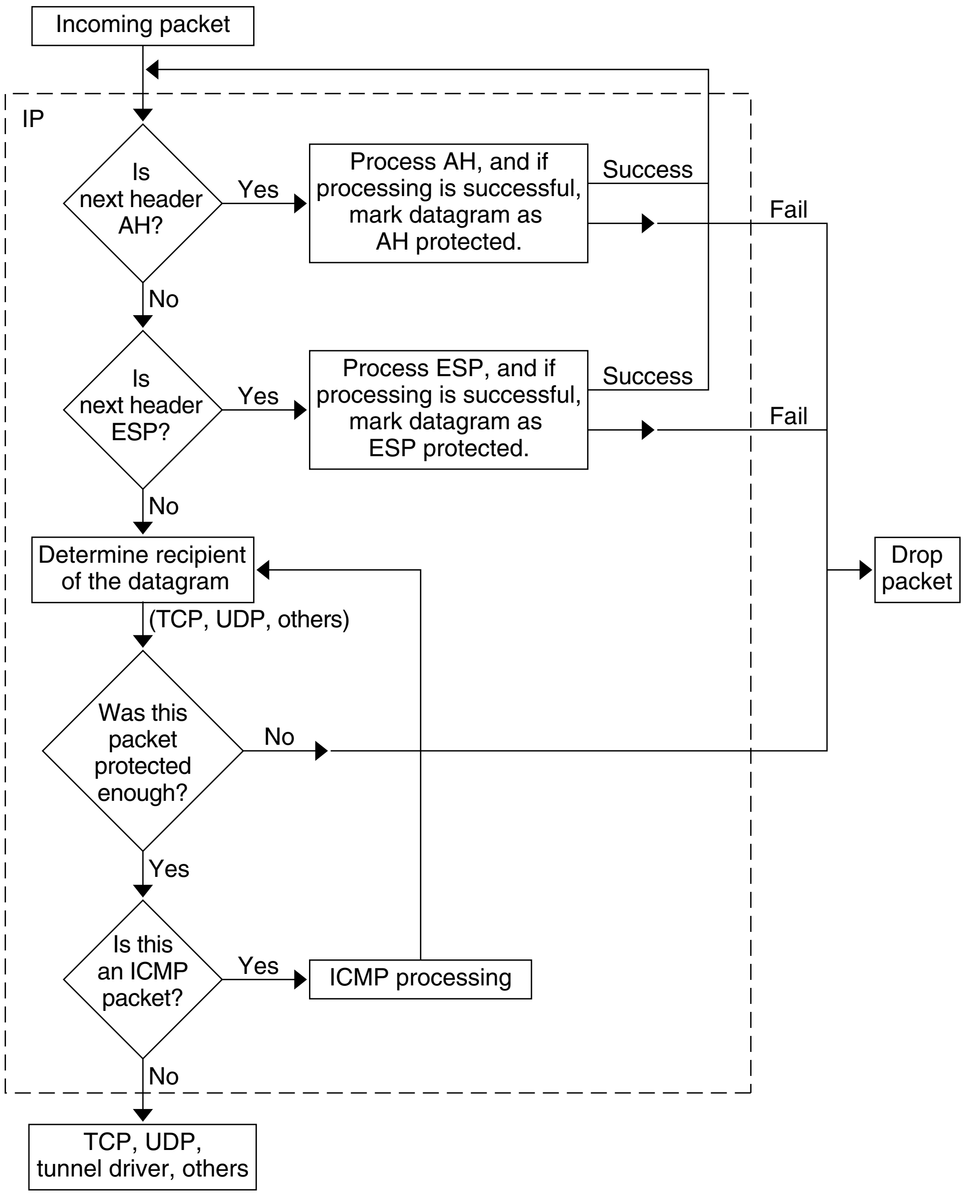


**FIGURE 6–2** IPsec Applied to Inbound Packet Process



# IPsec Security Associations

An IPsec *security association* (SA) specifies security properties that are recognized by communicating hosts. A single SA protects data in one direction. The protection is either to a single host or to a group (multicast) address. Because most communication is either peer-to-peer or client-server, two SAs must be present to secure traffic in both directions.

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, an arbitrary 32-bit value, is transmitted with an AH or ESP packet. The `ipsecah(7P)` and `ipsecesp(7P)` 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. For example, the IKE application and the `ipseckeys` command use the `PF_KEY` socket interface.

- For a more complete description of the IPsec SADB, see "Security Associations Database for IPsec" on page 120.
- For more information about how to manage the SADB, see the `pf_key(7P)` man page.

## Key Management in IPsec

Security associations (SAs) require keying material for authentication and for encryption. The managing of this keying material is called *key management*. The Internet Key Exchange (IKE) protocol handles key management automatically. You can also manage keys manually with the `ipseckey` command.

SAs on IPv4 and IPv6 packets can use either method of key management. Unless you have an overriding reason to use manual key management, IKE is preferred.

The Service Management Facility (SMF) feature of Oracle Solaris provides the following key management services for IPsec:

- `svc:/network/ipsec/ike:default` service – Is the SMF service for automatic key management. The `ike` service runs the `in.iked` daemon to provide automatic key management. For a description of IKE, see Chapter 9, "Internet Key Exchange (Overview)." For more information about the `in.iked` daemon, see the `in.iked(1M)` man page. For information about the `ike` service, see the "IKE Service" on page 163.
- `svc:/network/ipsec/manual-key:default` service – Is 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 "Utilities for SA Generation in IPsec" on page 121. For a detailed description of the `ipseckey` command options, see the `ipseckey(1M)` man page.

# IPsec Protection Mechanisms

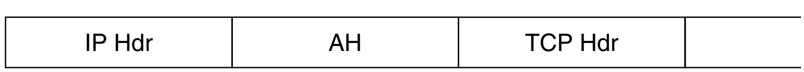
IPsec provides two security protocols for protecting data:

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

AH protects data with an authentication algorithm. An ESP protects data with an encryption algorithm. ESP can and should be used with an authentication mechanism. If you are not traversing a NAT, you can combine ESP with AH. Otherwise, you can use an authentication algorithm and an encryption mechanism with ESP. A combined mode algorithm, such as AES-GCM, provides encryption and authentication within a single algorithm.

## Authentication Header

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

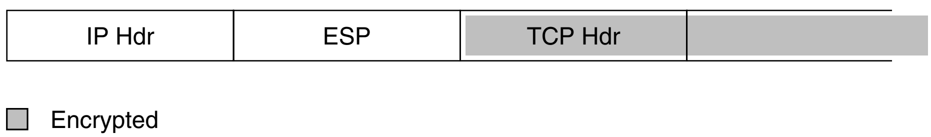



The transport header can be TCP, UDP, SCTP, or ICMP. If a tunnel is being used, the transport header can be another IP header.

# Encapsulating Security Payload

The encapsulating security payload (ESP) module provides confidentiality over what the ESP encapsulates. ESP also provides the services that AH provides. However, ESP only provides its protections over the part of the datagram that ESP encapsulates. ESP provides optional 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.

ESP encapsulates its data, so ESP only protects the data that follows its beginning in the datagram, as shown in the following illustration.




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

If self-encapsulation is set, a copy of the IP header is made to construct an IP-in-IP datagram. For example, when self-encapsulation is not set on a TCP socket, the datagram is sent in the following format:

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

When self-encapsulation is set on that TCP socket, the datagram 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 87.

## Security Considerations When Using AH and ESP

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

**TABLE 6–2** Protections Provided by AH and ESP in IPsec

| Protocol | Packet Coverage | Protection | Against Attacks |
|---|---|---|---|
| AH | Protects packet from the IP header to the transport header | 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 following the beginning of ESP in the datagram. | 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 protocols use two types of algorithms, authentication and encryption. The AH module uses authentication algorithms. The ESP module 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`(1M) man page. You can also use the functions that are described in the `getipsecalgbyname`(3NSL) man page to retrieve the properties of algorithms.

IPsec uses the Cryptographic Framework to access the algorithms. The Cryptographic Framework provides a central repository for algorithms, in addition to other services. The framework enables IPsec to take advantage of high performance cryptographic hardware accelerators.

For more information, see the following:

- Chapter 11, "Cryptographic Framework (Overview)," in *Oracle Solaris 11.1 Administration: Security Services*
- Chapter 8, "Introduction to the Oracle Solaris Cryptographic Framework," in *Developer's Guide to Oracle Solaris 11 Security*

## Authentication Algorithms in IPsec

Authentication algorithms produce an integrity checksum value or *digest* that is based on the data and a key. The AH module uses authentication algorithms. The ESP module can use authentication algorithms as well.

### Encryption Algorithms in IPsec

Encryption algorithms encrypt data with a key. The ESP module in IPsec uses encryption algorithms. The algorithms operate on data in units of a *block size*.

## IPsec Protection Policies

IPsec protection policies can use any of the security mechanisms. IPsec policies can be applied at the following levels:

- On a system-wide level
- On a per-socket level

IPsec applies the system-wide policy to outbound datagrams and inbound datagrams. Outbound datagrams are either sent with protection or without protection. If protection is applied, the algorithms are either specific or non-specific. You can apply some additional rules to outbound datagrams, because of the additional data that is known by the system. Inbound datagrams can be either accepted or dropped. The decision to drop or accept an inbound datagram is based on several criteria, which sometimes overlap or conflict. Conflicts are resolved by determining which rule is parsed first. The traffic is automatically accepted, except when a policy entry states that traffic should bypass all other policies.

The policy that normally protects a datagram can be bypassed. You can either specify an exception in the system-wide policy, or you can request a bypass in the per-socket policy. For traffic within a system, policies are enforced, but actual security mechanisms are not applied. Instead, the outbound policy on an intra-system packet translates into an inbound packet that has had those mechanisms applied.

You use the `ipsecinit.conf` file and the `ipsecconf` command to configure IPsec policies. For details and examples, see the `ipsecconf(1M)` man page.

## Transport and Tunnel Modes in IPsec

The IPsec standards define two distinct modes of IPsec operation, *transport mode* and *tunnel mode*. The modes do not affect the encoding of packets. The packets are protected by AH, ESP, or both in each mode. The modes differ in policy application when the inner packet is an IP packet, as follows:

- In transport mode, the outer header determines the IPsec policy that protects the inner IP packet.
- In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

In transport mode, the outer header, the next header, and any ports that the next header supports, can be used to determine IPsec policy. 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. Similarly, if the next header is an IP header, the outer header and the inner IP header can be used to determine IPsec policy.

Tunnel mode works only for IP-in-IP datagrams. Tunneling in tunnel mode can be useful when computer workers at home are connecting to a central computer location. In tunnel mode, IPsec policy is enforced on the contents of the inner IP datagram. Different IPsec policies 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. Unlike transport mode, in tunnel mode the outer IP header does not dictate the policy of its inner IP datagram.

Therefore, 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. However, if a dynamic routing protocol is run over a tunnel, do not use subnet selection or address selection because the view of the network topology on the peer network could change. Changes would invalidate the static IPsec policy. For examples of tunneling procedures that include configuring static routes, see "Protecting a VPN With IPsec" on page 101.

In Oracle Solaris, tunnel mode can be enforced only on an IP tunneling network interface. For information about tunneling interfaces, see Chapter 6, "Configuring IP Tunnels," in *Configuring and Administering Oracle Solaris 11.1 Networks*. The `ipsecconf` command 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.

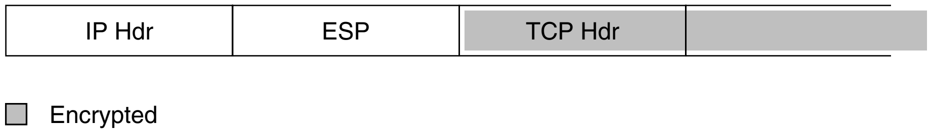
In transport mode, ESP, AH, or both, can protect the datagram.

The following figure shows an IP header with an unprotected TCP packet.

**FIGURE 6–3** Unprotected IP Packet Carrying TCP Information

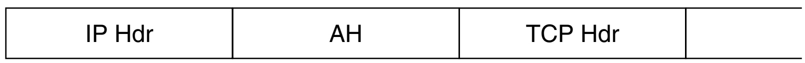

In transport mode, ESP protects the data as shown in the following figure. The shaded area shows the encrypted part of the packet.

**FIGURE 6–4** Protected IP Packet Carrying TCP Information




In transport mode, AH protects the data as shown in the following figure.

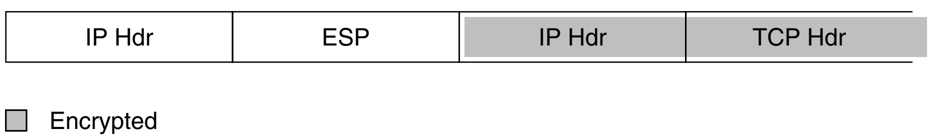
**FIGURE 6–5** Packet Protected by an Authentication Header




AH protection, even in transport mode, covers most of the IP header.

In tunnel mode, the entire datagram is *inside* the protection of an IPsec header. The datagram in Figure 6–3 is protected in tunnel mode by an outer IPsec header, and in this case ESP, as is shown in the following figure.

**FIGURE 6–6** IPsec Packet Protected in Tunnel Mode




The `ipsecconf` command includes keywords to set tunnels in tunnel mode or transport mode.

- For details on per-socket policy, see the `ipsec(7P)` man page.
- For an example of per-socket policy, see “How to Use IPsec to Protect a Web Server From Nonweb Traffic” on page 99.
- For more information about tunnels, see the `ipsecconf(1M)` man page.
- For an example of tunnel configuration, see “How to Protect a VPN With IPsec in Tunnel Mode” on page 104.

# Virtual Private Networks and IPsec

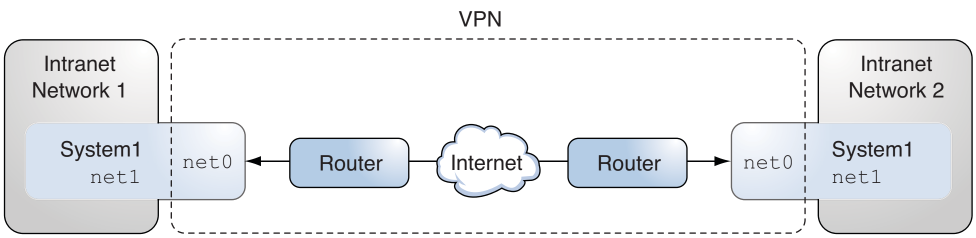
A configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see “How to Create and Configure an IP Tunnel” in *Configuring and Administering Oracle Solaris 11.1 Networks*.

A tunnel creates an apparent physical interface to IP. The physical link's integrity depends on the underlying security protocols. If you set up the security associations (SAs) securely, then you can trust the tunnel. Packets that exit the tunnel must have originated from the peer that was specified in the tunnel destination. If this trust exists, you can use per-interface IP forwarding to create a virtual private network (VPN).

You can add IPsec protections to a VPN. IPsec secures the connection. For example, an organization that uses VPN technology to connect offices with separate networks can add IPsec to secure the traffic between the two offices.

The following figure illustrates how two offices form a VPN with IPsec deployed on their network systems.

**FIGURE 6–7** Virtual Private Network




For a detailed example of the setup procedure, see "How to Protect a VPN With IPsec in Tunnel Mode" on page 104.

# 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.

NAT stands for network address translation. A NAT box is used to translate a private internal address into a unique Internet address. NATs are very common at public access points to the Internet, such as hotels. For a fuller discussion, see "Using IP Filter's NAT Feature" on page 42.

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 can be retrieved from `http://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

To use IPsec across a NAT, see “Configuring IKE for Mobile Systems (Task Map)” on page 153.

# 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 RFCs and “SCTP Protocol” in *System Administration Guide: IP Services*.

# IPsec and Oracle Solaris Zones

For shared-IP zones, IPsec is configured from the global zone. The IPsec policy configuration file, `ipsecinit.conf`, exists in the global zone only. The file can have entries that apply to non-global zones, as well as entries that apply to the global zone.

For exclusive-IP zones, IPsec is configured per non-global zone.

For information about how to use IPsec with zones, see "Protecting Traffic With IPsec" on page 95. For information about zones, see Chapter 15, "Introduction to Oracle Solaris Zones," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

# IPsec and Logical Domains

IPsec works with logical domains. The logical domain must be running a version of Oracle Solaris that includes IPsec, such as the Oracle Solaris 10 release.

To create logical domains, you must use the Oracle VM Server for SPARC, which was previously called Logical Domains. For information about how to configure logical domains, see *Oracle VM Server for SPARC 2.2 Administration Guide*.

# IPsec Utilities and Files

Table 6–3 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, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*.

- For instructions on implementing IPsec on your network, see "Protecting Traffic With IPsec" on page 95.
- For more details about IPsec utilities and files, see Chapter 8, "IP Security Architecture (Reference)."

**TABLE 6–3** List of Selected IPsec Utilities and Files

| IPsec Utility, File, or Service | Description | Man Page |
| --- | --- | --- |
| `svc:/network/ipsec/ipsecalgs` | The SMF service that manages IPsec algorithms. | `ipsecalgs(1M)` |
| `svc:/network/ipsec/manual-key` | The SMF service that manages manually keyed IPsec SAs. | `ipseckey(1M)` |
| `svc:/network/ipsec/policy` | The SMF service that manages IPsec policy. | `smf(5)`, `ipsecconf(1M)` |
| `svc:/network/ipsec/ike` | The SMF service for the automatic management of IPsec SAs by using IKE. | `smf(5)`, `in.iked(1M)` |
| `/etc/inet/ipsecinit.conf` file | IPsec policy file.<br>The SMF `policy` service uses this file to configure IPsec policy at system boot. | `ipsecconf(1M)` |

**TABLE 6–3** List of Selected IPsec Utilities and Files *(Continued)*

| IPsec Utility, File, or Service | Description | Man Page |
|---|---|---|
| `ipsecconf` command | IPsec policy command. Useful for viewing and modifying the current IPsec policy, and for testing.<br><br>Is used by the SMF `policy` service to configure IPsec policy at system boot. | `ipsecconf`(1M) |
| `PF_KEY` socket interface | Interface for the security associations database (SADB). Handles manual key management and automatic key management. | `pf_key`(7P) |
| `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`(1M) |
| `/etc/inet/secret/ipseckeys` file | Contains manually keyed SAs.<br><br>Is 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>Is used by the SMF `ipsecalgs` service to synchronize known IPsec algorithms with the kernel at system boot. | `ipsecalgs`(1M) |
| `/etc/inet/ipsecalgs` file | Contains the configured IPsec protocols and algorithm definitions. This file is managed by the `ipsecalgs` command and must never be edited manually. | |
| `/etc/inet/ike/config` file | IKE configuration and policy file. By default, this file does not exist. Key management is based on rules and global parameters in the `/etc/inet/ike/config` file. See "IKE Utilities and Files" on page 127.<br><br>If this file exists, the `svc:/network/ipsec/ike` service starts the IKE daemon, `in.iked`, to provide automatic key management. | `ike.config`(4) |

CHAPTER 7

# Configuring IPsec (Tasks)

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



For overview information about IPsec, see Chapter 6, "IP Security Architecture (Overview)." For reference information about IPsec, see Chapter 8, "IP Security Architecture (Reference)."

## Protecting Traffic With IPsec

This section provides procedures that 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 101. For additional procedures to manage IPsec and to use SMF commands with IPsec and IKE, see "Managing IPsec and IKE" on page 108.

The following information applies to all IPsec configuration tasks:

- **IPsec and zones –** To manage IPsec policy and keys for a shared-IP non-global zone, create the IPsec policy file in the global zone, and run the IPsec configuration commands from the global zone. Use the source address that corresponds to the non-global zone that is being configured. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.
- **IPsec and RBAC –** To use roles to administer IPsec, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *Oracle Solaris 11.1 Administration: Security Services*. For an example, see "How to Configure a Role for Network Security" on page 110.
- **IPsec and SCTP –** IPsec can be used to protect Streams Control Transmission Protocol (SCTP) associations, but caution must be used. For more information, see "IPsec and SCTP" on page 91.

- **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. Such a configuration enables you to tunnel IPv6 over an IPv4 network, for example.

The following task map points to procedures that set up IPsec between one or more systems. The `ipsecconf(1M)`, `ipseckey(1M)`, and `ipadm(1M)` man pages also describe useful procedures in their respective Examples sections.

| Task | Description | For Instructions |
|---|---|---|
| Secure traffic between two systems. | Protects packets from one system to another system. | "How to Secure Traffic Between Two Systems With IPsec" on page 96 |
| Secure a web server by using IPsec policy. | Requires non-web traffic to use IPsec. Web clients are identified by particular ports, which bypass IPsec checks. | "How to Use IPsec to Protect a Web Server From Nonweb Traffic" on page 99 |
| Display IPsec policies. | Displays the IPsec policies that are currently being enforced, in the order of enforcement. | "How to Display IPsec Policies" on page 100 |
| Use IKE to automatically create keying material for IPsec SAs. | Provides the raw data for security associations. | "Configuring IKE (Task Map)" on page 131 |
| Set up a secure virtual private network (VPN). | Sets up IPsec between two systems across the Internet. | "Protecting a VPN With IPsec" on page 101 |

## ▼ How to Secure Traffic Between Two Systems With IPsec

This procedure assumes the following setup:

- The two systems are named `enigma` and `partym`.
- Each system has an IP address. This can be an IPv4 address, an IPv6 address, or both.
- Each system requires ESP encryption with the AES algorithm, which requires a key of 128 bits, and ESP authentication with a SHA-2 message digest, which requires a key of 512 bits.
- 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** IPsec policy can be configured in the global zone or in an exclusive-IP stack zone. The policy for a shared-IP stack zone must be configured in the global zone. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.

To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile. To edit system files and create keys, you must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

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

   This step enables the Service Management Facility (SMF) to use the system names without depending on nonexistent naming services. For more information, see the `smf(5)` man page.

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

   ```
   # Secure communication with enigma
   192.168.116.16 enigma
   ```

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

   ```
   # Secure communication with partym
   192.168.13.213 partym
   ```

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.

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

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

   ```
   {laddr enigma raddr partym} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

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

   ```
   {laddr partym raddr enigma} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
   ```

   For the syntax of IPsec policy entries, see the `ipsecconf(1M)` man page.

**4 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 IKE (Task Map)" on page 131. For the syntax of the IKE configuration file, see the `ike.config(4)` man page.

---

**Note –** If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 108.

---

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

```
# ipsecconf -f -c /etc/inet/ipsecinit.conf
```

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

**6 Refresh the IPsec policy.**

```
# svcadm refresh svc:/network/ipsec/policy:default
```

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

```
# svcadm enable svc:/network/ipsec/policy:default
```

**7 Activate the keys for IPsec.**

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

  ```
  # svcadm enable svc:/network/ipsec/ike:default
  ```

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

  ```
  # svcadm restart svc:/network/ipsec/ike:default
  ```

If you manually configured keys in Step 4, complete "How to Manually Create IPsec Keys" on page 108 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 114.

## Example 7–1 Adding IPsec Policy When 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.

- First, the administrator configures the first system by performing Step 1 through Step 5 of the preceding procedure.
- Then, 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:
other-system #
```

- 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.
- Then, the administrator ends the ssh session.

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

- Finally, 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.

## ▼ How to Use IPsec to Protect a Web Server From Nonweb Traffic

A secure web server allows web clients to talk to the web service. On a secure web server, traffic that is not web traffic *must* pass security checks. The following procedure includes bypasses for web traffic. In addition, this web server can make unsecured DNS client requests. All other traffic requires ESP with AES and SHA-2 algorithms.

**Before You Begin** You must be in the global zone to configure IPsec policy. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.

You have completed "How to Secure Traffic Between Two Systems With IPsec" on page 96 so that the following conditions are in effect:

- Communication between the two systems is protected by IPsec.
- Keying material is being generated by IKE.
- You have verified that packets are being protected.

To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile. To edit system files, you must assume the root role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the ssh command for a secure remote login. For an example, see Example 7–1.

**1 Determine which services need to bypass security 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 `/etc/inet/ipsecinit.conf` file:

```
# 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 -f -c /etc/inet/ipsecinit.conf
```

**4 Refresh the IPsec policy.**

```
# svcadm refresh svc:/network/ipsec/policy:default
```

**5 Refresh the keys for IPsec.**

Restart the `ike` service.

```
# svcadm restart svc:/network/ipsec/ike
```

If you manually configured the keys, follow the instructions in "How to Manually Create IPsec Keys" on page 108.

Your setup is complete. Optionally, you can perform Step 6.

**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
#
{laddr webserver} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

Verify the syntax then refresh the IPsec policy to activate it.

```
remote-system # ipsecconf -f -c /etc/inet/ipsecinit.conf
remote-system # svcadm refresh svc:/network/ipsec/policy:default
```

A remote system can communicate securely with the web server for nonweb traffic only when the systems' IPsec policies match.

## ▼ How to Display IPsec Policies

You can see the policies that are configured in the system when you issue the `ipsecconf` command without any arguments.

**Before You Begin** You must run the `ipsecconf` command in the global zone. For an exclusive-IP zone, you run the `ipsecconf` command in the non-global zone.

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

- **Display IPsec policies.**
  - **Display the global IPsec policy entries in the order that the entries were added.**

    ```
    $ ipsecconf
    ```

    The command displays each entry with an *index* followed by a number.
  - **Display the IPsec policy entries in the order in which a match occurs.**

    ```
    $ ipsecconf -l -n
    ```
  - **Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.**

    ```
    $ ipsecconf -L -n
    ```

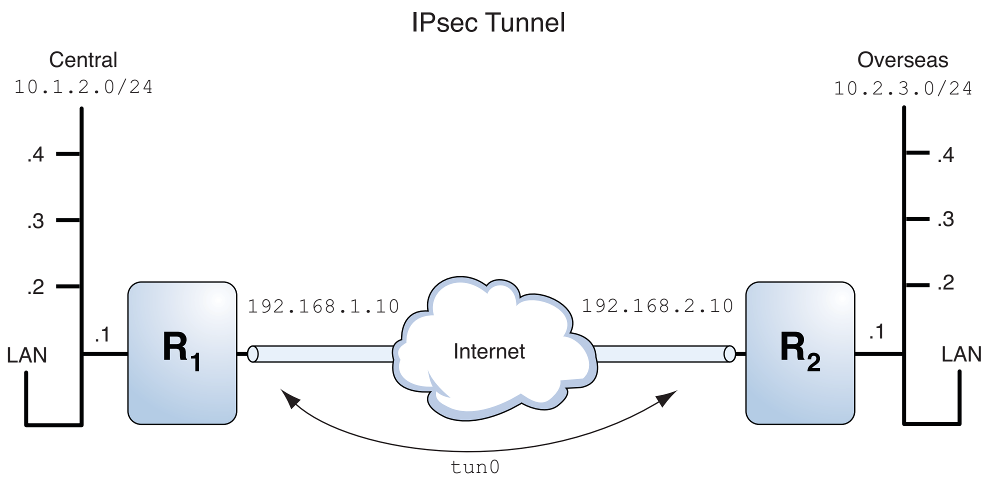
# Protecting a VPN With IPsec

Oracle Solaris can configure a VPN that is protected by IPsec. Tunnels can be created in *tunnel mode* or in *transport mode*. For a discussion, see "Transport and Tunnel Modes in IPsec" on page 87. 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 Traffic With IPsec" on page 95.

For examples of IPsec policies for tunnels in tunnel mode, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 101.

## Examples of Protecting a VPN With IPsec by Using Tunnel Mode

**FIGURE 7–1** Tunnel Protected by IPsec




The following examples assume that the tunnel is configured for all subnets of the LANs:

```
## Tunnel configuration ##
# Tunnel name is tun0
# Intranet point for the source is 10.1.2.1
# Intranet point for the destination is 10.2.3.1
# Tunnel source is 192.168.1.10
# Tunnel destination is 192.168.2.10

# Tunnel name address object is tun0/to-central
# Tunnel name address object is tun0/to-overseas
```

**EXAMPLE 7–2** Creating a Tunnel That All Subnets Can Use

In this example, all traffic from the local LANs of the Central LAN in Figure 7–1 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 7–3** Creating a Tunnel That Connects Two Subnets Only

In this example, only traffic between subnet `10.1.2.0/24` of the Central LAN and subnet `10.2.3.0/24` 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 10.1.2.0/24 raddr 10.2.3.0/24}
```

**EXAMPLE 7–3** Creating a Tunnel That Connects Two Subnets Only *(Continued)*

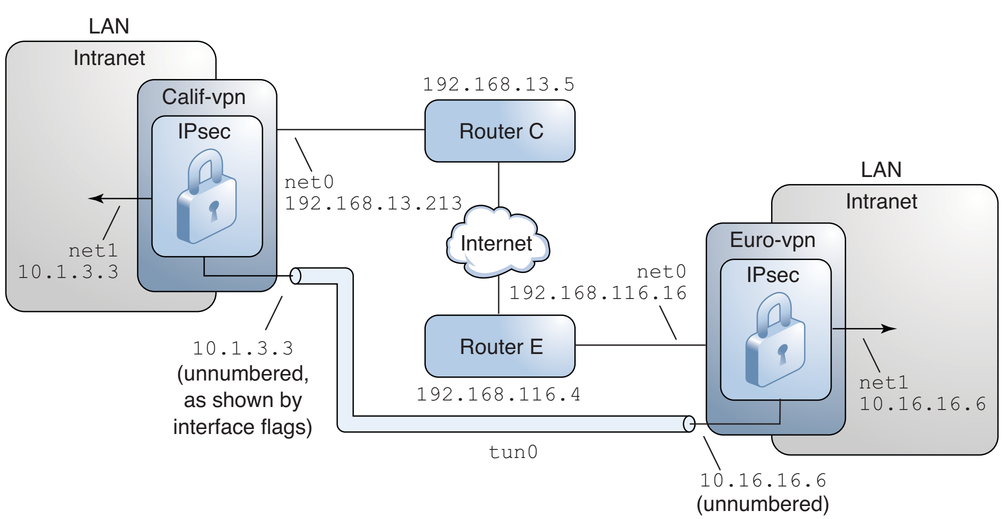
```
ipsec {encr_algs aes encr_auth_algs sha512 shared}
```

# Description of the Network Topology for the IPsec Tasks to Protect a VPN

The procedures that follow this section assume the following setup. For a depiction of the network, see Figure 7–2.

- Each system is using an IPv4 address space.
- Each system has two interfaces. The `net0` interface connects to the Internet. In this example, Internet IP addresses begin with `192.168`. The `net1` interface connects to the company's LAN, its intranet. In this example, intranet IP addresses begin with the number `10`.
- Each system requires ESP authentication with the SHA-2 algorithm. In this example, the SHA-2 algorithm requires a 512-bit key.
- Each system requires ESP encryption with the AES algorithm. The AES algorithm uses a 128-bit or 256-bit key.
- Each system can connect to a router that has direct access to the Internet.
- Each system uses shared security associations.

**FIGURE 7–2** Sample VPN Between Offices Connected Across the Internet




As the preceding illustration shows, the procedures use the following configuration parameters.

| Parameter | Europe | California |
| --- | --- | --- |
| System name | euro-vpn | calif-vpn |
| System intranet interface | net1 | net1 |
| System intranet address, also the *-point* address in Step 6 | 10.16.16.6 | 10.1.3.3 |
| System intranet address object | net1/inside | net1/inside |
| System Internet interface | net0 | net0 |
| System Internet address, also the *tsrc* address in Step 6 | 192.168.116.16 | 192.168.13.213 |
| Name of Internet router | router-E | router-C |
| Address of Internet router | 192.168.116.4 | 192.168.13.5 |
| Tunnel name | tun0 | tun0 |
| Tunnel name address object | tun0/v4tunaddr | tun0/v4tunaddr |

For information about tunnel names, see "Tunnel Configuration and Administration With the dladm Command" in *Configuring and Administering Oracle Solaris 11.1 Networks*. For information about address objects, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1* and the `ipadm(1M)` man page.

## ▼ How to Protect a VPN 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 Traffic Between Two Systems With IPsec" on page 96. The setup is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 103.

For a fuller description of the reasons for running particular commands, see the corresponding steps in "How to Secure Traffic Between Two Systems With IPsec" on page 96.

**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** You must be in the global zone to configure IPsec policy for the system or for a shared-IP zone. For an exclusive-IP zone, you configure IPsec policy in the non-global zone.

To run configuration commands, you must become an administrator who is assigned the Network Management and Network IPsec Management rights profiles. To edit system files, you must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**1 Control the flow of packets before configuring IPsec.**

**a. Disable IP forwarding and IP dynamic routing.**

```
# routeadm -d ipv4-routing
# ipadm set-prop -p forwarding=off ipv4
# routeadm -u
```

Turning off IP forwarding prevents packets from being forwarded from one network to another network through this system. For a description of the `routeadm` command, see the `routeadm(1M)` man page.

**b. Turn on IP strict multihoming.**

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

Turning on 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 loopback mounts and the `ssh` service are running.

```
# svcs | grep network
online         Aug_02   svc:/network/loopback:default
...
online         Aug_09   svc:/network/ssh:default
```

**2 Add IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. For additional examples, see "Examples of Protecting a VPN With IPsec by Using Tunnel Mode" on page 101.

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, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.16.16.6 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, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.1.3.3 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 IKE (Task Map)" on page 131. For the syntax of the IKE configuration file, see the `ike.config(4)` man page.

---

**Note –** If you must generate and maintain your keys manually, see "How to Manually Create IPsec Keys" on page 108.

---

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

```
# ipsecconf -f -c /etc/inet/ipsecinit.conf
```

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

5 **Refresh the IPsec policy.**

```
# svcadm refresh svc:/network/ipsec/policy:default
```

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

```
# svcadm enable svc:/network/ipsec/policy:default
```

6 **Create and configure the tunnel,** *tunnel-name***.**

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.**

If the interface `net1` does not already exist, the first command creates it.

```
# ipadm create-addr -T static -a local=10.1.3.3 net1/inside
# dladm create-iptun -T ipv4 -a local=10.1.3.3,remote=10.16.16.6 tun0
```

```
# ipadm create-addr -T static \
-a local=192.168.13.213,remote=192.168.116.16 tun0/v4tunaddr
```

b. **On the `euro-vpn` system, create the tunnel and configure it.**

```
# ipadm create-addr -T static -a local=10.16.16.6 net1/inside
# dladm create-iptun -T ipv4 -a local=10.16.16.6,remote=10.1.3.3 tun0
# ipadm create-addr -T static \
-a local=192.168.116.16,remote=192.168.13.213 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`(1M) and `ipadm`(1M) man pages, and "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*. For information about customized names, see "Network Devices and Datalink Names" in *Oracle Solaris Administration: Network Interfaces and Network Virtualization*.

7 **On each system, configure forwarding.**

```
# ipadm set-ifprop -m ipv4 -p forwarding=on net1
# 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 turned on.

Because the `net1` interface is *inside* the intranet, IP forwarding must be turned on for `net1`. Because `tun0` connects the two systems through the Internet, IP forwarding must remain on for `tun0`. The `net0` interface has its IP forwarding turned off to prevent an *outside* adversary from injecting packets into the protected intranet. The *outside* refers to the Internet.

8 **On each system, prevent the advertising of the private interface.**

```
# ipadm set-addrprop -p private=on net0
```

Even if `net0` has IP forwarding turned off, 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 192.168.13.5
```

**b. On the euro-vpn system, add the following route:**

```
# route -p add net default  192.168.116.4
```

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(1M) and in.routed(1M) man pages.

# Managing IPsec and IKE

The following task map points to tasks that you might use when managing IPsec.

| Task | Description | For Instructions |
| --- | --- | --- |
| Create or replace security associations manually. | Provides the raw data for security associations:<br>■ IPsec algorithm name and keying material<br>■ The security parameter index (SPI)<br>■ IP source and destination addresses, and other parameters | "How to Manually Create IPsec Keys" on page 108 |
| 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 110 |
| Manage IPsec and keying material as a set of SMF services. | Describes when and how to use the commands that enable, disable, refresh, and restart services. Also describes the commands that change the property values of services. | "How to Manage IPsec and IKE Services" on page 112 |
| Check that IPsec is protecting the packets. | Examines snoop output for specific headers that indicate how the IP datagrams are protected. | "How to Verify That Packets Are Protected With IPsec" on page 114 |

## ▼ How to Manually Create IPsec Keys

The following procedure provides the keying material for Step 4 in "How to Secure Traffic Between Two Systems With IPsec" on page 96. You are generating keys for two systems, partym and enigma. You generate the keys on one system, and then use the keys from the first system on both systems.

**Before You Begin** You must be in the global zone to manually manage keying material for a non-global zone.

You must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Generate the keying material for the SAs.**

   a. **Determine the keys that you require.**

   You need three hexadecimal random numbers for outbound traffic and three hexadecimal random numbers for inbound traffic. Therefore, one system needs to generate the following numbers:

   - Two hexadecimal random numbers as the value for the `spi` keyword. 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 algorithm for AH. Each number must be 512 characters long. One number is for `dst enigma`. One number is for `dst partym`.
   - Two hexadecimal random numbers for the 3DES algorithm for ESP. Each number must be 168 characters long. One number is for `dst enigma`. One number is for `dst partym`.

   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 *Oracle Solaris 11.1 Administration: Security Services* and the IPsec example in that section.

2. **In the `root` role on each system, add the keys to the manual keys file for IPsec.**

   a. **Edit the `/etc/inet/secret/ipseckeys` file on the `enigma` system to appear similar to the following:**

```
# ipseckeys - This file takes the file format documented in
#   ipseckey(1m).
#   Note that naming services might not be available when this file
#   loads, just like ipsecinit.conf.
#
#   Backslashes indicate command continuation.
#
# for outbound packets on enigma
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3des \
   auth_alg sha512  \
   encrkey  d41fb74470271826a8e7a80d343cc5aa... \
   authkey  e896f8df7f78d6cab36c94ccf293f031...
#
# for inbound packets
add esp spi 0x122a43e4 \
```

```
src 192.168.13.213 dst 192.168.116.16 \
encr_alg 3des \
auth_alg sha512  \
encrkey dd325c5c137fb4739a55c9b3a1747baa... \
authkey ad9ced7ad5f255c9a8605fba5eb4d2fd...
```

b. **Protect the file with read-only permissions.**

```
# chmod 400 /etc/inet/secret/ipseckeys
```

c. **Verify the syntax of the file.**

```
# ipseckey -c -f /etc/inet/secret/ipseckeys
```

---

**Note –** The keying material on the two systems *must* be identical.

---

3 **Activate the keys for IPsec.**

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

```
# svcadm enable svc:/network/ipsec/manual-key:default
```

- **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.

## ▼ How to Configure a Role for Network Security

If you are using the role-based access control (RBAC) feature of Oracle Solaris to administer your systems, you 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
Console User:RO::Manage System as the Console User...
Network Management:RO::Manage the host and network configuration...
Network Autoconf Admin:RO::Manage Network Auto-Magic configuration via nwamd...
Network Autoconf User:RO::Network Auto-Magic User...
Network ILB:RO::Manage ILB configuration via ilbadm...
Network LLDP:RO::Manage LLDP agents via lldpadm...
Network VRRP:RO::Manage VRRP instances...
Network Observability:RO::Allow access to observability devices...
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.**

```
% getent exec_attr | grep "Network Management"
...
Network Management:solaris:cmd:::/sbin/dlstat:euid=dladm;egid=sys
...
Network Management:solaris:cmd:::/usr/sbin/snoop:privs=net_observability
Network Management:solaris:cmd:::/usr/sbin/spray:euid=0 ...
```

**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.

**4 Create a network security role that includes the Network Management rights profile.**

A role with the Network Security or the Network IPsec Management rights profile, in addition to the Network Management profile, can execute the `ipadm`, `ipseckey`, and `snoop` commands, among others, with appropriate privilege.

To create the role, assign the role to a user, and register the changes with the naming service, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

## Example 7–4 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:

- The administrator names the first role LinkWifi.
  - The administrator assigns the Network Wifi, Network Link Security, and Network Management rights profiles to the role.
  - Then, the administrator assigns the LinkWifi role to the appropriate users.
- The administrator names the second role IPsec Administrator.
  - The administrator assigns the Network IPsec Management and the Network Management rights profiles to the role.
  - Then, the administrator assigns the IPsec Administrator role to the appropriate users.

## ▼ How to Manage IPsec and IKE Services

The following steps provide the most likely uses of the SMF services for IPsec, IKE, and manual key management. By default, the `policy` and `ipsecalgs` services are enabled. Also by default, the `ike` and `manual-key` services are disabled.

**Before You Begin** You must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 To manage IPsec policy, do one of the following:**

- **After adding new policies to the `ipsecinit.conf` file, refresh the `policy` service.**

```
# svcadm refresh svc:/network/ipsec/policy
```

- **After changing the value of a service property, view the property value, then refresh and restart the `policy` service.**

```
# svccfg -s policy setprop config/config_file=/etc/inet/MyIpsecinit.conf
# svccfg -s policy listprop config/config_file
config/config_file  astring  /etc/inet/MyIpsecinit.conf
# svcadm refresh svc:/network/ipsec/policy
# svcadm restart svc:/network/ipsec/policy
```

**2 To automatically manage keys, do one of the following:**

- **After adding entries to the `/etc/inet/ike/config` file, enable the `ike` service.**

```
# svcadm enable svc:/network/ipsec/ike
```

- **After changing entries in the `/etc/inet/ike/config` file, restart the `ike` service.**

```
# svcadm restart svc:/network/ipsec/ike:default
```

- **After changing the value of a service property, view the property value, then refresh and restart the service.**

  ```
  # svccfg -s ike setprop config/admin_privilege = astring: "modkeys"
  # svccfg -s ike listprop config/admin_privilege
  config/admin_privilege  astring  modkeys
  # svcadm refresh svc:/network/ipsec/ike
  # svcadm restart svc:/network/ipsec/ike
  ```

- **To stop the `ike` service, disable it.**

  ```
  # svcadm disable svc:/network/ipsec/ike
  ```

3 **To manually manage keys, do one of the following:**

- **After adding entries to the `/etc/inet/secret/ipseckeys` file, enable the `manual-key` service.**

  ```
  # svcadm enable svc:/network/ipsec/manual-key:default
  ```

- **After changing the `ipseckeys` file, refresh the service.**

  ```
  # svcadm refresh manual-key
  ```

- **After changing the value of a service property, view the property value, then refresh and restart the service.**

  ```
  # svccfg -s manual-key setprop config/config_file=/etc/inet/secret/MyIpseckeyfile
  # svccfg -s manual-key listprop config/config_file
  config/config_file  astring  /etc/inet/secret/MyIpseckeyfile
  # svcadm refresh svc:/network/ipsec/manual-key
  # svcadm restart svc:/network/ipsec/manual-key
  ```

- **To prevent manual key management, disable the `manual-key` service.**

  ```
  # svcadm disable svc:/network/ipsec/manual-key
  ```

4 **If you modify the IPsec protocols and algorithms table, refresh the `ipsecalgs` service.**

```
# svcadm refresh svc:/network/ipsec/ipsecalgs
```

**Troubleshooting** Use the `svcs` *service* command to find the status of a service. If the service is in `maintenance` mode, follow the debugging suggestions in the output of the `svcs -x` *service* command.

# 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 AH: if you used auth_alg to protect the traffic.
- ESP: Prefix indicates that encrypted data is being sent. You see ESP: 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 "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **On one system, such as partym, assume the root role.**

```
% su -
Password:       Type root password
#
```

2. **From the partym system, prepare to snoop packets from a remote system.**

   In a terminal window on partym, snoop the packets from the enigma system.

```
# snoop -d net0 -v enigma
Using device /dev/bge (promiscuous mode)
```

3. **Send a packet from the remote system.**

   In another terminal window, remotely log in to the enigma system. Provide your password. Then, assume the root role and send a packet from the enigma system to the partym system. The packet should be captured by the snoop -v enigma command.

```
% ssh enigma
Password:       Type your password
% su -
Password:       Type root password
# ping partym
```

4. **Examine the snoop output.**

   On the partym system, 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 = 192.168.116.16, enigma
IP:   Destination address = 192.168.13.213, partym
```

```
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

# IP Security Architecture (Reference)

This chapter contains the following reference information:



For instructions on how to implement IPsec on your network, see Chapter 7, "Configuring IPsec (Tasks)." For an overview of IPsec, see Chapter 6, "IP Security Architecture (Overview)."

## 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 "IKE Service" on page 163.

For information about SMF, see Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*. Also see the `smf`(5), `svcadm`(1M), and `svccfg`(1M) 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 datagrams. Forwarded datagrams are not subjected to policy checks that are added by using this command. The ipsecconf command also configures the security policy database (SPD). For IPsec policy options, see the ipsecconf(1M) man page.

You must assume the root role to invoke the ipsecconf command. The command accepts entries that protect traffic in both directions. The command also accepts 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 Traffic With IPsec" on page 95.

# ipsecinit.conf 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(1M) 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 10.0.0.1 is the IPv4 address of this host (laddr)
# and 10.0.0.2 is the IPv4 address of the remote host (raddr).
```

```
#

{laddr 10.0.0.1 raddr 10.0.0.2} 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(1M) man page for warnings on using symbolic names and
# many more examples, configuration options and supported algorithms.
#
# This example assumes that 10.0.0.1 is the IPv4 address of this host (laddr)
# and 10.0.0.2 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 10.0.0.1 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 does 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`(3SOCKET) and `accept`(3SOCKET) man pages. If you are in doubt, restart the connection.

Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:

- Your source address is a host that can be looked up over the network.
- Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the `ipsecconf` command. Use `ssh`, or a console or other hard-connected TTY for the safest mode of operation.

# 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. However, if the file needs to be modified, use the ipsecalgs command. The file must never be edited directly. 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 RFC 2407. In a general sense, a DOI defines data formats, network traffic exchange types, and conventions for naming security-relevant information. Security policies, cryptographic algorithms, and cryptographic modes are examples of security-relevant information.

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(1M) man page. For more information about the Cryptographic Framework, see Chapter 11, "Cryptographic Framework (Overview)," in *Oracle Solaris 11.1 Administration: Security Services*.

# 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 SADBs are maintained by a user process, or possibly multiple cooperating processes, that send messages over a special kind of socket. This method of maintaining SADBs is analogous to the method that is described in the route(7P) man page. Only the root role can access the database.

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(7P) man page.

# Utilities for SA Generation in IPsec

The IKE protocol provides automatic key management for IPv4 and IPv6 addresses. See Chapter 10, "Configuring IKE (Tasks)," for instructions on how to set up IKE. The manual keying utility is the `ipseckey` command, which is described in the `ipseckey(1M)` man page.

You use the `ipseckey` command to manually populate the security associations database (SADB). 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.

The `ipseckey` command can be used to view all SAs that are known to the system, whether the keys were added manually or by IKE. With the `-c` option, the `ipseckey` command checks the syntax of the keys file that you provide as an argument.

IPsec SAs that are added by the `ipseckey` command are not persistent over system reboot. To enable manually added SAs at system boot, add entries to the `/etc/inet/secret/ipseckeys` file, then enable the `svc:/network/ipsec/manual-key:default` service. For the procedure, see "How to Manually Create IPsec Keys" on page 108.

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 `pf_key(7P)` 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, not manual keying with `ipseckey`, if possible.

---

You should consider the following issues when you handle keying material and use the `ipseckey` command:

- Have you refreshed the keying material? Periodic key refreshment is a fundamental security practice. Key refreshment guards against potential weaknesses of the algorithm and keys, and limits the damage of an exposed key.
- Is the TTY going over a network? Is the `ipseckey` command in interactive mode?
  - In interactive mode, the security of the keying material is the security of the network path for this TTY's traffic. You should avoid using the `ipseckey` command over a clear-text telnet or rlogin session.

  - Even local windows might be vulnerable to attacks by a concealed program that reads window events.
- Have you used the -f option? Is the file being accessed over the network? Can the file be read by the world?
  - An adversary can read a network-mounted file as the file is being read. You should avoid using a world-readable file that contains keying material.
  - Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:
    - Your source address is a host that can be looked up over the network.
    - Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the ipseckey command. Use ssh, or a console or other hard-connected TTY for the safest mode of operation.

# 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. Therefore, 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(1M) 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 114.

Third-party network analyzers are also available, such as the free open-source software Wireshark (http://www.wireshark.org/about.html), which is bundled with this release.

CHAPTER 9

# Internet Key Exchange (Overview)

Internet Key Exchange (IKE) automates key management for IPsec. Oracle Solaris implements IKEv1. This chapter contains the following information about IKE:



For instructions on implementing IKE, see Chapter 10, "Configuring IKE (Tasks)." For reference information, see Chapter 11, "Internet Key Exchange (Reference)." For information about IPsec, see Chapter 6, "IP Security Architecture (Overview)."

## Key Management With 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 version 1 (IKE) to automate key management. IKE easily scales to provide a secure channel for a large volume of traffic. IPsec SAs on IPv4 and IPv6 packets can take advantage of IKE.

IKE can take advantage of available hardware acceleration and hardware storage. Hardware accelerators permit intensive key operations to be handled off the system. Key storage on hardware provides an additional layer of protection.

# IKE Key Negotiation

The IKE daemon, `in.iked`, negotiates and authenticates keying material for IPsec SAs in a secure manner. The daemon uses random seeds for keys from internal functions provided by the OS. IKE 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(1M)` man page.

## IKE Key Terminology

The following table lists terms that are used in key negotiation, provides their commonly used acronyms, and gives a definition and use for each term.

**TABLE 9–1** Key Negotiation Terms, Acronyms, and Uses

| Key Negotiation Term | Acronym | Definition and Use |
|---|---|---|
| Key exchange | | The process of generating keys for asymmetric cryptographic algorithms. The two main methods are the RSA and the Diffie-Hellman protocols. |
| Diffie-Hellman algorithm | DH | A key exchange algorithm that provides key generation and key authentication. Often called *authenticated key exchange*. |
| RSA algorithm | RSA | A key exchange algorithm that provides key generation and key transport. The protocol is named for its three creators, Rivest, Shamir, and Adleman. |
| Perfect forward secrecy | PFS | Applies to authenticated key exchange only. 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. |
| Oakley group | | A method for establishing keys for Phase 2 in a secure manner. The Oakley group is used to negotiate PFS. See Section 6 of The Internet Key Exchange (IKE) (`http://www.faqs.org/rfcs/rfc2409.html`). |

## IKE Phase 1 Exchange

The Phase 1 exchange is known as *Main Mode*. In the Phase 1 exchange, IKE uses public key encryption methods to authenticate itself with peer IKE entities. The result is an Internet Security Association and Key Management Protocol (ISAKMP) security association (SA). An ISAKMP SA is a secure channel for IKE to negotiate keying material for the IP datagrams. Unlike IPsec SAs, the ISAKMP SAs are bidirectional, so only one security association is needed.

How IKE negotiates keying material in the Phase 1 exchange is configurable. IKE 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 used
- The interfaces that are affected
- The security protocols and their algorithms
- The authentication method

The two authentication methods are preshared keys and public key certificates. The public key certificates can be self-signed. Or, the certificates can be issued by a certificate authority (CA) from a public key infrastructure (PKI) organization.

## IKE Phase 2 Exchange

The Phase 2 exchange is known as *Quick Mode*. In the Phase 2 exchange, IKE creates and manages the IPsec SAs between systems that are running the IKE daemon. IKE uses the secure channel that was created in the Phase 1 exchange to protect the transmission of keying material. The IKE daemon creates the keys from a random number generator by using the `/dev/random` device. The daemon refreshes 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`.

# IKE Configuration Choices

The `/etc/inet/ike/config` configuration file contains IKE policy entries. For two IKE daemons to authenticate each other, the entries must be valid. Also, keying material must be available. The entries in the configuration file determine the method for using the keying material to authenticate the Phase 1 exchange. The choices are preshared keys or public key certificates.

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. Public key certificates can be self-signed, or the certificates can be installed from a PKI organization. For more information, see the `ike.config(4)` man page.

## IKE With Preshared Key Authentication

Preshared keys are used to authenticate two or more peer systems. The preshared key is a hexadecimal number or ASCII string that is created by an administrator on one system. The key is then shared out of band with administrators of the peer systems in a secure way. If the preshared key is intercepted by an adversary, that adversary might be able to impersonate one of the peer systems.

The preshared key on the peers that use this authentication method must be identical. The 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 the `ike.preshared(4)` man page.

## IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Public keys use the Diffie-Hellman algorithm (DH) for authenticating and negotiating keys. Public key certificates come in two flavors. The certificates can be self-signed, or the certificates can be certified by a certificate authority (CA).

Self-signed public key certificates are created by you, the administrator. The `ikecert certlocal -ks` command creates the private part of the public-private key pair for the system. You then get the self-signed certificate output in X.509 format from the remote system. The remote system's certificate is input to the `ikecert certdb` command for the public part of the key pair. The self-signed certificates reside in the `/etc/inet/ike/publickeys` directory on the communicating systems. When you use the `-T` option, the certificates reside on attached hardware.

Self-signed certificates are a halfway point between preshared keys and CAs. Unlike preshared keys, a self-signed certificate can be used on a mobile machine or on a system that might be renumbered. To self-sign a certificate for a system without a fixed number, use a `DNS` (`www.example.org`) or `email` (`root@domain.org`) alternative name.

Public keys can be delivered by a PKI or a CA organization. You install the public keys and their accompanying CAs in the `/etc/inet/ike/publickeys` directory. When you use the `-T` option, the certificates reside on attached hardware. Vendors also issue certificate revocation lists (CRLs). Along with installing the keys and CAs, you are responsible for installing the CRL in the `/etc/inet/ike/crls` directory.

CAs have the advantage of being certified by an outside organization, rather than by the site administrator. In a sense, CAs are notarized certificates. As with self-signed certificates, CAs can be used on a mobile machine or on a system that might be renumbered. Unlike self-signed certificates, CAs can very easily scale to protect a large number of communicating systems.

# IKE Utilities and Files

The following table summarizes the configuration files for IKE policy, the storage locations for IKE keys, and the various commands and services that implement IKE. For more about services, see Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*.

**TABLE 9–2** IKE Configuration Files, Key Storage Locations, Commands, and Services

| File, Location, Command, or Service | Description | Man Page |
| --- | --- | --- |
| `svc:/network/ipsec/ike` | The SMF service that manages IKE. | `smf`(5) |
| `/usr/lib/inet/in.iked` | Internet Key Exchange (IKE) daemon. Activates automated key management when the `ike` service is enabled. | `in.iked`(1M) |
| `/usr/sbin/ikeadm` | 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`(1M) |
| `/usr/sbin/ikecert` | Certificate database management command for manipulating local databases that hold public key certificates. The databases can also be stored on attached hardware. | `ikecert`(1M) |
| `/etc/inet/ike/config` | Default configuration file for the IKE policy. Contains the site's rules for matching inbound IKE requests and preparing outbound IKE requests.<br><br>If this file exists, the `in.iked` daemon starts when the `ike` service is enabled. The location of this file can be changed by the `svccfg` command. | `ike.config`(4) |
| `ike.preshared` | Preshared keys file in the `/etc/inet/secret` directory. Contains secret keying material for authentication in the Phase 1 exchange. Used when configuring IKE with preshared keys. | `ike.preshared`(4) |
| `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`(1M) |
| `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`(1M) |
| `crls` directory | Directory in the `/etc/inet/ike` directory that holds revocation lists for public keys and certificate files. | `ikecert`(1M) |

**TABLE 9–2** IKE Configuration Files, Key Storage Locations, Commands, and Services *(Continued)*

| File, Location, Command, or Service | Description | Man Page |
|---|---|---|
| Sun Crypto Accelerator 6000 board | Hardware that accelerates public key operations by offloading the operations from the operating system. The board also stores public keys, private keys, and public key certificates. The Sun Crypto Accelerator 6000 board is a FIPS 140-2 certified device at Level 3. | ikecert(1M) |

CHAPTER 10

# Configuring IKE (Tasks)

This chapter describes how to configure the Internet Key Exchange (IKE) for your systems. After IKE is configured, it automatically generates keying material for IPsec on your network. This chapter contains the following information:



For overview information about IKE, see Chapter 9, "Internet Key Exchange (Overview)." For reference information about IKE, see Chapter 11, "Internet Key Exchange (Reference)." For more procedures, see the Examples sections of the `ikeadm`(1M), `ikecert`(1M), and `ike.config`(4) man pages.

## Displaying IKE Information

You can view the algorithms and groups that can be used in Phase 1 IKE negotiations.

### ▼ How to Display Available Groups and Algorithms for Phase 1 IKE Exchanges

In this procedure, you determine which Diffie-Hellman groups are available for use in Phase 1 IKE exchanges. You also view the encryption and authentication algorithms that are available for IKE Phase 1 exchanges. The numeric values match the values that are specified for these algorithms by the Internet Assigned Numbers Authority (IANA).

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

1. **Display the list of Diffie-Hellman groups that IKE can use in Phase 1.**

   Diffie-Hellman groups set up IKE SAs.

```
# ikeadm dump groups
Value Strength Description
1     66       ietf-ike-grp-modp-768
2     77       ietf-ike-grp-modp-1024
5     91       ietf-ike-grp-modp-1536
14    110      ietf-ike-grp-modp-2048
15    130      ietf-ike-grp-modp-3072
16    150      ietf-ike-grp-modp-4096
17    170      ietf-ike-grp-modp-6144
18    190      ietf-ike-grp-modp-8192

Completed dump of groups
```

   You would use one of these values as the argument to the `oakley_group` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha encr_alg aes }
```

2. **Display the list of authentication algorithms that IKE can use in Phase 1.**

```
# ikeadm dump authalgs
Value Name
1     md5
2     sha1
4     sha256
5     sha384
6     sha512

Completed dump of authalgs
```

   You would use one of these names as the argument to the `auth_alg` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha256 encr_alg 3des }
```

3. **Display the list of encryption algorithms that IKE can use in Phase 1.**

```
# ikeadm dump encralgs
Value Name
3     blowfish-cbc
5     3des-cbc
1     des-cbc
7     aes-cbc

Completed dump of encralgs
```

You would use one of these names as the argument to the `encr_alg` parameter in an IKE Phase 1 transform, as in:

```
p1_xform
  { auth_method preshared oakley_group 15 auth_alg sha256 encr_alg aes }
```

**See Also** For tasks to configure IKE rules that require these values, see "Configuring IKE (Task Map)" on page 131.

# Configuring IKE (Task Map)

You can use preshared keys, self-signed certificates, and certificates from a Certificate Authority (CA) to authenticate IKE. A rule links the particular IKE authentication method with the end points that are being protected. Therefore, you can use one or all IKE authentication methods on a system. A pointer to a PKCS #11 library enables IKE to use an attached hardware accelerator.

After configuring IKE, complete the IPsec task that uses the IKE configuration. The following table refers you to task maps that focus on a specific IKE configuration.

| Task | Description | For Instructions |
|---|---|---|
| Configure IKE with preshared keys. | Protects communications between two systems by having the systems share a secret key. | "Configuring IKE With Preshared Keys (Task Map)" on page 131 |
| Configure IKE with public key certificates. | Protects communications with public key certificates. The certificates can be self-signed, or they can be vouched for by a PKI organization. | "Configuring IKE With Public Key Certificates (Task Map)" on page 136 |
| Cross a NAT boundary. | Configures IPsec and IKE to communicate with a mobile system | "Configuring IKE for Mobile Systems (Task Map)" on page 153 |
| Configure IKE to use a hardware keystore to generate a certificate pair. | Enables a Sun Crypto Accelerator 6000 board to accelerate IKE operations and to store public key certificates. | "Configuring IKE to Find Attached Hardware" on page 160 |

# Configuring IKE With Preshared Keys (Task Map)

The following table points to procedures to configure and maintain IKE with preshared keys.

| Task | Description | For Instructions |
|---|---|---|
| Configure IKE with preshared keys. | Creates an IKE configuration file and one key to be shared. | "How to Configure IKE With Preshared Keys" on page 132 |

| Task | Description | For Instructions |
|---|---|---|
| Add preshared keys to a running IKE system. | Adds a new IKE policy entry and new keying material to a system that is currently enforcing IKE policy. | "How to Update IKE for a New Peer System" on page 135 |

# Configuring IKE With Preshared Keys

Preshared keys is the simplest authentication method for IKE. If you are configuring peer system to use IKE, and you are the administrator of these systems, using preshared keys is a good choice. However, unlike public key certificates, preshared keys are tied to IP addresses. You can associate preshared keys with specific IP addresses or ranges of IP addresses. Preshared keys cannot be used with mobile systems or systems that might be renumbered, unless the renumbering is within the specified range of IP addresses.

## ▼ How to Configure IKE 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 `enigma` and `partym`. Substitute the names of your systems for the names `enigma` and `partym`.

---

**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, in addition to the `solaris.admin.edit/etc/inet/ike/config` authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**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 IKE configuration examples work with the `ipsecinit.conf` examples in "How to Secure Traffic Between Two Systems With IPsec" on page 96.

a. **For example, modify the `/etc/inet/ike/config` file on the `enigma` system:**

```
### ike/config file on enigma, 192.168.116.16

## Global parameters
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
p2_pfs 2
#
## The rule to communicate with partym
#  Label must be unique
{ label "enigma-partym"
  local_addr 192.168.116.16
  remote_addr 192.168.13.213
  p1_xform
   { auth_method preshared oakley_group 5 auth_alg sha256 encr_alg aes }
  p2_pfs 5
}
```

b. **Modify the `/etc/inet/ike/config` file on the `partym` system:**

```
### ike/config file on partym, 192.168.13.213
## Global Parameters
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg 3des }
p2_pfs 2

## The rule to communicate with enigma
#  Label must be unique
{ label "partym-enigma"
  local_addr 192.168.13.213
  remote_addr 192.168.116.16
p1_xform
 { auth_method preshared oakley_group 5 auth_alg sha256 encr_alg aes }
p2_pfs 5
}
```

3 **On each system, verify the syntax of the file.**

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

**4 Create the file `/etc/inet/secret/ike.preshared` on each system.**

Put the preshared key in each file.

**a. For example, on the `enigma` system, the `ike.preshared` file would appear similar to the following:**

```
# ike.preshared on enigma, 192.168.116.16
#...
{ localidtype IP
    localid 192.168.116.16
    remoteidtype IP
    remoteid 192.168.13.213
    # The preshared key can also be represented in hex
# as in 0xf47cb0f432e14480951095f82b
# key "This is an ASCII Cqret phrAz, use str0ng p@ssword tekniques"
}
```

**b. On the `partym` system, the `ike.preshared` file would appear similar to the following:**

```
# ike.preshared on partym, 192.168.13.213
#...
{ localidtype IP
    localid 192.168.13.213
    remoteidtype IP
    remoteid 192.168.116.16
    # The preshared key can also be represented in hex
# as in 0xf47cb0f432e14480951095f82b
    key "This is an ASCII Cqret phrAz, use str0ng p@ssword tekniques"
    }
```

**5 Enable the IKE service.**

```
# svcadm enable ipsec/ike
```

**Example 10–1** Refreshing an IKE Preshared Key

When IKE administrators want to refresh the preshared key, they edit the files on the peer systems and restart the `in.iked` daemon.

First, the administrator adds a preshared key entry, valid for any host on the `192.168.13.0/24` subnet.

```
#...
{ localidtype IP
    localid 192.168.116.0/24
    remoteidtype IP
    remoteid 192.168.13.0/24
    # enigma and partym’s shared passphrase for keying material
key "LOooong key Th@t m^st Be Ch*angEd \’reguLarLy)"
    }
```

Then, the administrator restarts the IKE service on every system.

```
# 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.

## How to Update IKE 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 IKE.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKE 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, in addition to the `solaris.admin.edit/etc/inet/ike/config` authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

1. **Create a rule for IKE to manage the keys for the new system that is using IPsec.**

   a. **For example, on the `enigma` system, add the following rule to the `/etc/inet/ike/config` file:**

```
### ike/config file on enigma, 192.168.116.16

## The rule to communicate with ada

{label "enigma-to-ada"
 local_addr 192.168.116.16
 remote_addr 192.168.15.7
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg sha256 encr_alg aes}
 p2_pfs 5
    }
```

   b. **On the `ada` system, add the following rule:**

```
### ike/config file on ada, 192.168.15.7

## The rule to communicate with enigma

{label "ada-to-enigma"
 local_addr 192.168.15.7
 remote_addr 192.168.116.16
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg sha256 encr_alg aes}
 p2_pfs 5
}
```

**2 Create an IKE preshared key for the peer systems.**

a. **On the `enigma` system, add the following information to the `/etc/inet/secret/ike.preshared` file:**

```
# ike.preshared on enigma for the ada interface
#
{ localidtype IP
  localid 192.168.116.16
  remoteidtype IP
  remoteid 192.168.15.7
  # enigma and ada's shared key
  key "Twas brillig and the slivey toves did *s0mEtHiNg* be CareFULL hEEEr"
}
```

b. **On the ada system, add the following information to the `ike.preshared` file:**

```
# ike.preshared on ada for the enigma interface
#
{ localidtype IP
  localid 192.168.15.7
  remoteidtype IP
  remoteid 192.168.116.16
  # ada and enigma's shared key
  key "Twas brillig and the slivey toves did *s0mEtHiNg* be CareFULL hEEEr"
}
```

**3 On each system, refresh the `ike` service.**

```
# svcadm refresh ike
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

# Configuring IKE With Public Key Certificates (Task Map)

The following table provides pointers to procedures for creating public key certificates for IKE. The procedures include how to accelerate and store the certificates on attached hardware.

A public certificate must be unique, so the creator of a public key certificate generates an arbitrary, unique name for the certificate. Typically, an X.509 distinguished name is used. An alternate name can also be used for identification. The format of these names is *tag*=*value*. The values are arbitrary, though the format of the value must correspond to its tag type. For example, the format of the `email` tag is *name@domain.suffix*.

| Task | Description | For Instructions |
| --- | --- | --- |
| Configure IKE with self-signed public key certificates. | Creates and places two certificates on each system:<br>▪ A self-signed certificate<br>▪ The public key certificate from the peer system | "How to Configure IKE With Self-Signed Public Key Certificates" on page 137 |

| Task | Description | For Instructions |
| --- | --- | --- |
| Configure IKE with a PKI Certificate Authority. | Creates a certificate request, and then places three certificates on each system:<br>■ The certificate that the Certificate Authority (CA) creates from your request<br>■ The public key certificate from the CA<br>■ The CRL from the CA | "How to Configure IKE With Certificates Signed by a CA" on page 142 |
| Configure public key certificates in local hardware. | Involves one of:<br>■ Generating a self-signed certificate in the local hardware, then adding the public key from a remote system to the hardware.<br>■ Generating a certificate request in the local hardware, then adding the public key certificates from the CA to the hardware. | "How to Generate and Store Public Key Certificates in Hardware" on page 147 |
| Update the certificate revocation list (CRL) from a PKI. | Accesses the CRL from a central distribution point. | "How to Handle a Certificate Revocation List" on page 151 |

---

**Note –** To label packets and IKE negotiations on a Trusted Extensions system, follow the procedures in "Configuring Labeled IPsec (Task Map)" 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.

---

# Configuring IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Unlike preshared keys, a public key certificate can be used on a mobile machine or on a system that might be renumbered.

Public key certificates can also be generated and stored in attached hardware. For the procedure, see "Configuring IKE to Find Attached Hardware" on page 160.

## ▼ How to Configure IKE With Self-Signed Public Key Certificates

In this procedure, you create a certificate pair. The private key is stored on disk in the local certificate database and can be referenced by using the `certlocal` subcommand. The public portion of the certificate pair is stored in the public certificate database. It can be referenced by using the `certdb` subcommand. You exchange the public portion with a peer system. The combination of the two certificates is used to authenticate the IKE 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 out of band.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile, in addition to the `solaris.admin.edit/etc/inet/ike/config` authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**1 Create a self-signed certificate in the `ike.privatekeys` database.**

```
# ikecert certlocal -ks -m keysize -t keytype \
-D dname -A altname \
[-S validity-start-time] [-F validity-end-time] [-T token-ID]
```

| | |
|---|---|
| `-ks` | Creates a self-signed certificate. |
| `-m` *keysize* | Is the size of the key. The *keysize* can be 512, 1024, 2048, 3072, or 4096. |
| `-t` *keytype* | Specifies the type of algorithm to use. The *keytype* can be `rsa-sha1`, `rsa-md5`, or `dsa-sha1`. |
| `-D` *dname* | Is the X.509 distinguished name for the certificate subject. The *dname* typically has the form: C=*country*, O=*organization*, OU=*organizational unit*, CN=*common name*. Valid tags are C, O, OU, and CN. |
| `-A` *altname* | Is the alternate name for the certificate. The *altname* is in the form of `tag=value`. Valid tags are `IP`, `DNS`, `email`, and `DN`. |
| `-S` *validity-start-time* | Provides an absolute or relative valid start time for the certificate. |
| `-F` *validity-end-time* | Provides an absolute or relative valid end time for the certificate. |
| `-T` *token-ID* | Enables a PKCS #11 hardware token to generate the keys. The certificates are then stored in the hardware. |

**a. For example, the command on the `partym` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 2048 -t rsa-sha1 \
-D "O=exampleco, OU=IT, C=US, CN=partym" \
-A IP=192.168.13.213
Creating private key.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
```

```
MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
a...+
zBGi4QkNdI3f
-----END X509 CERTIFICATE-----
```

---

**Note –** The values of the `-D` and `-A` options are arbitrary. The values are used to identify the certificate only. They are not used to identify a system, such as 192.168.13.213. In fact, because these values are idiosyncratic, you must verify out of band that the correct certificate is installed on the peer systems.

---

b. **The command on the `enigma` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 2048 -t rsa-sha1 \
-D "O=exampleco, OU=IT, C=US, CN=enigma" \
-A IP=192.168.116.16
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. 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 `partym` certificate to the `enigma` administrator.**

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEfdZgKjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
a...+
zBGi4QkNdI3f
-----END X509 CERTIFICATE------
```

b. **The `enigma` administrator would send you the public portion of the `enigma` certificate.**

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.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 readable 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 `partym`.**

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.

```
partym # 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=partym>     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
partym # 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-sha1
        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 `partym` system has a valid certificate pair.

**b. Verify that the `enigma` system has `partym`'s public certificate.**

You can read the public key hash over the telephone.

Compare the hashes from Note 3 on `partym` in the preceding step with Note 4 on `enigma`.

```
enigma # 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=enigma>
        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=partym>
        Key Size: 2048
        Public key hash: 80829EC52FC5BA910F4764076C20FDCF     Note 4
```

The public key hash and subject name of the last certificate stored in `enigma`'s public certificate database match the private certificate for `partym` 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 `partym` 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=enigma"

# 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 "192.168.116.16"

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha256 encr_alg 3des }
p2_pfs 5

{
 label "US-partym to JA-enigmax"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=partym"
 remote_id "O=exampleco, OU=IT, C=US, CN=enigma"
```

```
 local_addr  192.168.13.213
# 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 `enigma` system, add `enigma` values for local parameters in the `ike/config` file.**

For the remote parameters, use `partym` values. Ensure that the value for the `label` keyword is unique on the local system.

```
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "O=exampleco, OU=IT, C=US, CN=enigma"
 remote_id "O=exampleco, OU=IT, C=US, CN=partym"

 local_addr  192.168.116.16
 remote_addr 0.0.0.0/0
...
```

6 **On the peer systems, enable IKE.**

```
partym # svcadm enable ipsec/ike
```

```
enigma # 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.

## ▼ How to Configure IKE With Certificates Signed by a CA

Public certificates from a Certificate Authority (CA) require negotiation with an outside organization. The certificates very easily scale to protect a large number of communicating systems.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile, in addition to the `solaris.admin.edit/etc/inet/ike/config` authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**1 Use the `ikecert certlocal -kc` command to create a certificate request.**

For a description of the arguments to the command, see Step 1 in "How to Configure IKE With Self-Signed Public Key Certificates" on page 137.

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname
```

**a. For example, the following command creates a certificate request on the `partym` system:**

```
# ikecert certlocal -kc -m 2048 -t rsa-sha1 \
> -D "C=US, O=PartyCompany\, Inc., OU=US-Partym, CN=Partym" \
> -A "DN=C=US, O=PartyCompany\, Inc., OU=US-Partym"
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 certificate request on the `enigma` system:**

```
# ikecert certlocal -kc -m 2048 -t rsa-sha1 \
> -D "C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax, CN=Enigmax" \
> -A "DN=C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax"
Creating software private keys.
...
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
8qlqdjaStLGfhDOO
-----END CERTIFICATE REQUEST-----
```

**2 Submit the certificate request to a PKI organization.**

The PKI organization can tell you how to submit the certificate request. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your certificate request into the form. When your request has been checked by the organization, the organization issues you the following two certificate objects and a list of revoked certificates:

- Your public key certificate – This certificate is based on the request that you submitted to the organization. The request that you submitted is part of this public key certificate. The certificate uniquely identifies you.
- A Certificate Authority – The organization's signature. The CA verifies that your public key certificate is legitimate.

- A Certificate Revocation List (CRL) – The latest list of certificates that the organization has revoked. The CRL is not sent separately as a certificate object if access to the CRL is embedded in the public key certificate.

  When a URI for the CRL is embedded in the public key certificate, IKE can automatically retrieve the CRL for you. Similarly, when a DN (directory name on an LDAP server) entry is embedded in the public key certificate, IKE can retrieve and cache the CRL from an LDAP server that you specify.

  See "How to Handle a Certificate Revocation List" on page 151 for an example of an embedded URI and an embedded DN entry in a public key certificate.

**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 126.

**a. Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**b. Add the public key certificate that you received from the PKI organization.**

```
# ikecert certdb -a < /tmp/PKIcert.eml
```

**c. Add the CA from the PKI organization.**

```
# ikecert certdb -a < /tmp/PKIca.eml
```

**d. If the PKI organization 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
<Control>-D
```

4 **Use the `cert_root` keyword to identify the PKI organization in the `/etc/inet/ike/config` file.**

Use the name that the PKI organization provides.

a. **For example, the `ike/config` file on the `partym` system might appear similar to the following:**

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform
 { auth_method rsa_sig oakley_group 1 auth_alg sha384 encr_alg aes}
p2_pfs 2

{
 label "US-partym to JA-enigmax - Example PKI"
 local_id_type dn
 local_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 local_addr  192.168.13.213
 remote_addr 192.168.116.16

 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 `enigma` system, create a similar file.**

Specifically, the `enigma` `ike/config` file should do the following:

- Include the same `cert_root` value.
- Use `enigma` values for local parameters.
- Use `partym` 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=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id   "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
```

```
 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

**5 Tell IKE how to handle CRLs.**

Choose the appropriate option:

- **No CRL available**

  If the PKI organization does not provide a CRL, add the keyword `ignore_crls` to the `ike/config` file.

  ```
  # Trusted root cert
  ...
  cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example,...
  ignore_crls
  ...
  ```

  The `ignore_crls` keyword tells IKE not to search for CRLs.

- **CRL available**

  If the PKI organization provides a central distribution point for CRLs, you can modify the `ike/config` file to point to that location.

  See "How to Handle a Certificate Revocation List" on page 151 for examples.

**Example 10–2** Using rsa_encrypt When Configuring IKE

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.

   For example, the `partym` administrator would send the following email:

   ```
   To: admin@ja.enigmaexample.com
   From: admin@us.partyexample.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MII...
   ----END X509 CERTIFICATE-----
   ```

   The `enigma` administrator would send the following email:

   ```
   To: admin@us.partyexample.com
   From: admin@ja.enigmaexample.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, IKE cannot retrieve the peer's certificate. As a result, the `rsa_encrypt` method requires that the IKE 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 three certificates for each communicating pair of systems:

- Your public key certificate
- The CA certificate
- The peer's public key certificate

**Troubleshooting –** The IKE payload, which includes the 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.

## ▼ How to Generate and Store Public Key Certificates in Hardware

Generating and storing public key certificates on hardware is similar to generating and storing public key certificates on your system. On hardware, the `ikecert certlocal` and `ikecert certdb` commands must identify the hardware. The `-T` option with the token ID identifies the hardware to the commands.

**Before You Begin**

- The hardware must be configured.
- The hardware uses the `/usr/lib/libpkcs11.so` library, unless the `pkcs11_path` keyword in the `/etc/inet/ike/config` file points to a different library. The library must be implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki), that is, a PKCS #11 library.

  See "How to Configure IKE to Find the Sun Crypto Accelerator 6000 Board" on page 160 for setup instructions.

You must become an administrator who is assigned the Network IPsec Management rights profile, in addition to the `solaris.admin.edit/etc/inet/ike/config` authorization. The `root` role has all of these rights. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

**1 Generate a self-signed certificate or a certificate request, and specify the token ID.**

Choose one of the following options:

---

**Note** – The Sun Crypto Accelerator 6000 board supports keys up to 2048 bits for RSA. For DSA, this board supports keys up to 1024 bits.

---

- **For a self-signed certificate, use this syntax.**

  ```
  # ikecert certlocal -ks -m 2048 -t rsa-sha1 \
  > -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
  > -a -T dca0-accel-stor IP=192.168.116.16
  Creating hardware private keys.
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  The argument to the `-T` option is the token ID from the attached Sun Crypto Accelerator 6000 board.

- **For a certificate request, use this syntax.**

  ```
  # ikecert certlocal -kc -m 2048 -t rsa-sha1 \
  > -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
  > -a -T dca0-accel-stor IP=192.168.116.16
  Creating hardware private keys.
  Enter PIN for PKCS#11 token:      Type user:password
  ```

For a description of the arguments to the `ikecert` command, see the `ikecert(1M)` man page.

**2 At the prompt for a PIN, type the Sun Crypto Accelerator 6000 user, a colon, and the user's password.**

If the Sun Crypto Accelerator 6000 board has a user `ikemgr` whose password is `rgm4tigt`, you would type the following:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
```

---

**Note** – The PIN response is stored on disk *as clear text*.

---

After you type the password, the certificate prints out:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
oKUDBbZ9O/pLWYGr
-----END X509 CERTIFICATE-----
```

**3 Send your certificate for use by the other party.**

Choose one of the following options:

- **Send the self-signed certificate to the remote system.**

  You can paste the certificate into an email.

- **Send the certificate request to an organization that handles PKI.**

  Follow the instructions of the PKI organization to submit the certificate request. For a more detailed discussion, see Step 2 of "How to Configure IKE With Certificates Signed by a CA" on page 142.

**4 On your system, edit the `/etc/inet/ike/config` file to recognize the certificates.**

Choose one of the following options:

- **Self-signed certificate**

  Use the values that the administrator of the remote system provides for the `cert_trust`, `remote_id`, and `remote_addr` parameters. For example, on the `enigma` system, the `ike/config` file would appear similar to the following:

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

cert_trust "192.168.116.16"       Local system's certificate Subject Alt Name
cert_trust "192.168.13.213"       Remote system's certificate Subject Alt name

...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

- **Certificate request**

  Type the name that the PKI organization provides as the value for the `cert_root` keyword. For example, the `ike/config` file on the `enigma` system might appear similar to the following:

```
# Trusted root cert
# This certificate is from Example PKI
```

```
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

5 **Place the certificates from the other party in the hardware.**

Respond to the PIN request as you responded in Step 2.

---

**Note –** You *must* add the public key certificates to the same attached hardware that generated your private key.

---

- **Self-signed certificate.**

  Add the remote system's self-signed certificate. In this example, the certificate is stored in the file, `DCA.ACCEL.STOR.CERT`.

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  If the self-signed certificate used `rsa_encrypt` as the value for the `auth_method` parameter, add the peer's certificate to the hardware store.

- **Certificates from a PKI organization.**

  Add the certificate that the organization generated from your certificate request, and add the certificate authority (CA).

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password

  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CA.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  To add a certificate revocation list (CRL) from the PKI organization, see "How to Handle a Certificate Revocation List" on page 151.

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

# How to Handle a Certificate Revocation List

A certificate revocation list (CRL) contains outdated or compromised certificates from a Certificate Authority. You have four ways to handle CRLs.

- You must instruct IKE to ignore CRLs if your CA organization does not issue CRLs. This option is shown in Step 5 in "How to Configure IKE With Certificates Signed by a CA" on page 142.
- You can instruct IKE to access the CRLs from a URI (uniform resource indicator) whose address is embedded in the public key certificate from the CA.
- You can instruct IKE 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 10–3.

The following procedure describes how to instruct IKE to use CRLs from a central distribution point.

**Before You Begin** You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

**1 Display the certificate that you received from the CA.**

`# ikecert certdb -lv` *certspec*

| | |
|---|---|
| `-l` | Lists certificates in the IKE certificate database. |
| `-v` | Lists the certificates in verbose mode. Use this option with care. |
| *certspec* | Is a pattern that matches a certificate in the IKE certificate database. |

For example, the following certificate was issued by Oracle. Details have been altered.

```
# ikecert certdb -lv example-protect.oracle.com
Certificate Slot Name: 0   Type: dsa-sha1
   (Private key in certlocal slot 0)
 Subject Name: <O=Oracle, CN=example-protect.oracle.com>
 Issuer Name: <CN=Oracle CA (Cl B), O=Oracle>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2011 Sep 19th, 21:11:11 GMT
      Not Valid After:  2015 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 = example-protect.oracle.com
```

```
      Key Usage: DigitalSignature KeyEncipherment
      [CRITICAL]
   CRL Distribution Points:
      Full Name:
         URI = #Ihttp://www.oracle.com/pki/pkismica.crl#i
         DN = <CN=Oracle CA (Cl B), O=Oracle>
      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 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 own web proxy
  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.oracle.com:389,ldap2.oracle.com"
  ...
  ```

IKE retrieves the CRL and caches the CRL until the certificate expires.

**Example 10–3** Pasting a CRL Into the Local certrldb Database

If the PKI organization's CRL is not available from a central distribution point, you can add the CRL manually to the local `certrldb` database. Follow the PKI organization'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 < Oracle.Cert.CRL
```

# Configuring IKE for Mobile Systems (Task Map)

The following table points to procedures to configure IKE to handle systems that log in remotely to a central site.

| 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 IKE for Off-Site Systems" on page 154 |
| Use a CA's public certificate and IKE 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 10–4 |
| Use a CA's public certificate and IKE 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 10–5 |
| Use self-signed certificates and IKE 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 10–6 |
| Use self-signed certificates and IKE 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 10–7 |

# Configuring IKE for Mobile Systems

When configured properly, home offices and mobile laptops can use IPsec and IKE to communicate with their company's central computers. A blanket IPsec policy that is combined with a public key authentication method enables off-site systems to protect their traffic to a central system.

# How to Configure IKE for Off-Site 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 PKI enable mobile systems to communicate with the central site.

**Before You Begin** You must assume the `root` role. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*. If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

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 IKE 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.domain.org,ldap2.domain.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.domain.org"
```

```
#cert_trust    "DNS=mobile.domain.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust    "email=root@central.domain.org"
#cert_trust    "email=user1@mobile.domain.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 5

p1_xform
{auth_method rsa_sig oakley_group 5 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.

```
# /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 IKE 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.

```
## /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.domain.org,ldap2.domain.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.domain.org"
#cert_trust    "DNS=central.domain.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
#cert_trust    "email=user1@domain.org"
#cert_trust    "email=root@central.domain.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.domain.org"
    local_addr 0.0.0.0/0

# Find central and trust the root certificate
    remote_id "central.domain.org"
    remote_addr 192.xxx.xxx.x

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256 }
}
```

3 **Enable the `ike` service.**

```
# svcadm enable svc:/network/ipsec/ike
```

### Example 10–4 Configuring a Central Computer to Accept IPsec 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 10–5.

```
## /etc/hosts on main1
main1 192.168.0.100

## /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.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
#
# Rule for off-site systems with root certificate
{
  label "Off-site system with root certificate"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# 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 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256}
}
```

### Example 10–5 Configuring a System Behind a NAT With IPsec

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 10–4.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100
```

```
## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} 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.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
#
# Rule for off-site systems with root certificate
{
  label "Off-site mobile1 with root certificate"
  local_id_type DNS
  local_id "mobile1.domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the root certificate
  remote_id "main1.domain.org"
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256 }
}
```

**Example 10–6** 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 10–7.

```
## /etc/hosts on main1
main1 192.168.0.100

## /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.domain.org"
```

```
cert_trust    "jdoe@domain.org"
cert_trust    "user2@domain.org"
cert_trust    "user3@domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site systems with trusted certificates"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# 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 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg aes auth_alg sha256 }
}
```

**Example 10–7** 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 10–6.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100


## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} 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@domain.org"
cert_trust    "DNS=main1.domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site mobile1 with trusted certificate"
  local_id_type email
  local_id "jdoe@domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the certificate
  remote_id "main1.domain.org"
```

```
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 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.

# Configuring IKE to Find Attached Hardware

Public key certificates can also be stored on attached hardware. The Sun Crypto Accelerator 6000 board provides storage, and enables public key operations to be offloaded from the system to the board.

## ▼ How to Configure IKE to Find the Sun Crypto Accelerator 6000 Board

**Before You Begin** The following procedure assumes that a Sun Crypto Accelerator 6000 board is attached to the system. The procedure also assumes that the software for the board has been installed and that the software has been configured. For instructions, see *Sun Crypto Accelerator 6000 Board Version 1.1 User's Guide* (`http://download.oracle.com/docs/cd/E19321-01/820-4144-12/820-4144-12.pdf`).

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

If you log in remotely, use the `ssh` command for a secure remote login. For an example, see Example 7–1.

1 **Check that the PKCS #11 library is linked.**

IKE uses the library's routines to handle key generation and key storage on the Sun Crypto Accelerator 6000 board. Type the following command to determine whether a PKCS #11 library has been linked:

```
$ ikeadm get stats
...
PKCS#11 library linked in from /usr/lib/libpkcs11.so
$
```

2 **Find the token ID for the attached Sun Crypto Accelerator 6000 board.**

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":
```

```
"Sun Metaslot                    "
```

The library returns a token ID, also called a keystore name, of 32 characters. In this example, you could use the `Sun Metaslot` token with the `ikecert` commands to store and accelerate IKE keys.

For instructions on how to use the token, see "How to Generate and Store Public Key Certificates in Hardware" on page 147.

The trailing spaces are automatically padded by the `ikecert` command.

**Example 10–8** Finding and Using Metaslot Tokens

Tokens can be stored on disk, on an attached board, or in the softtoken keystore that the Cryptographic Framework provides. The softtoken keystore token ID might resemble the following.

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":

"Sun Metaslot                    "
```

To create a passphrase for the softtoken keystore, see the `pktool(1)` man page.

A command that resembles the following would add a certificate to the softtoken keystore. `Sun.Metaslot.cert` is a file that contains the CA certificate.

```
# ikecert certdb -a -T "Sun Metaslot" < Sun.Metaslot.cert
Enter PIN for PKCS#11 token:      Type user:passphrase
```

**Next Steps** If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy.

CHAPTER 11

# Internet Key Exchange (Reference)

This chapter contains the following reference information about IKE:



For instructions on implementing IKE, see Chapter 10, "Configuring IKE (Tasks)." For overview information, see Chapter 9, "Internet Key Exchange (Overview)."

## IKE Service

`svc:/network/ipsec/ike:default` **service** – The Service Management Facility (SMF) provides the `ike` service to manage IKE. By default, this service is disabled. Before enabling this service, you must create an IKE configuration file, `/etc/inet/ike/config`.

The following `ike` service properties are configurable:

- `config_file` **property** – Is the location of the IKE configuration file. The initial value is `/etc/inet/ike/config`.
- `debug_level` **property** – Is 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`(1M) man page.
- `admin_privilege` **property** – Is the level of privilege of the `in.iked` daemon. The initial value is `base`. Other values are `modkeys` and `keymat`. For details, see "`ikeadm` Command" on page 165.

For information about SMF, see Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*. Also see the `smf(5)`, `svcadm(1M)`, and `svccfg(1M)` man pages.

# IKE Daemon

The `in.iked` 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 communicate securely.

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` service, the `in.iked` daemon runs at system boot.

When the IKE daemon runs, the system authenticates itself to its peer IKE entity in the Phase 1 exchange. The peer is defined in the IKE policy file, as are the authentication methods. The daemon then establishes the keys for the Phase 2 exchange. At an interval specified in the policy file, the IKE keys are refreshed automatically. The `in.iked` 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(7P)` man page.

Two commands support the IKE daemon. The `ikeadm` command can be used to view and temporarily modify the IKE policy. To permanently modify the IKE policy, you modify properties of the `ike` service. To modify properties of the IKE service, see "How to Manage IPsec and IKE Services" on page 112. The `ikeadm` command can also be used to view Phase 1 SAs, policy rules, preshared keys, available Diffie-Hellman groups, Phase 1 encryption and authentication algorithms, and the certificate cache.

The `ikecert` command enables you to view and manage the public key databases. This command manages the local databases, `ike.privatekeys` and `publickeys`. This command also manages public key operations and the storage of public keys on hardware.

# IKE Configuration File

The IKE configuration file, `/etc/inet/ike/config`, manages the keys for the interfaces 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 path to an attached hardware accelerator. For examples of IKE policy files, see "Configuring IKE With Preshared Keys (Task Map)" on page 131. For examples and descriptions of IKE policy entries, see the `ike.config(4)` man page.

The IPsec SAs that IKE supports protect the IP datagrams according to the policies in the IPsec configuration file, /etc/inet/ipsecinit.conf. The IKE policy file determines if perfect forward security (PFS) is used when creating the IPsec SAs.

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). IKE uses this PKCS #11 library to access hardware for key acceleration and key storage.

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 119.

# 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(1M) 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 keying material. The base level is the default level of privilege.

modkeys level — You can remove, change, and add preshared keys.

keymat level — You can view the actual keying material with the ikeadm command.

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 procedure, see "How to Manage IPsec and IKE Services" on page 112.

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 121.

# IKE 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 Internet Security Association and Key Management Protocol (ISAKMP) SAs. The `ipseckeys` file contains the preshared keys for IPsec SAs. The files are protected at `0600`. The `secret` directory is protected at `0700`.

- You create an `ike.preshared` file when you configure the `ike/config` file to require preshared keys. You enter keying material for ISAKMP SAs, that is, for IKE authentication, in the `ike.preshared` file. Because the preshared keys are used to authenticate the Phase 1 exchange, the file must be valid before the `in.iked` daemon starts.
- The `ipseckeys` file contains keying material for IPsec SAs. For examples of manually managing the file, see "How to Manually Create IPsec Keys" on page 108. The IKE daemon does not use this file. The keying material that IKE generates for IPsec SAs is stored in the kernel.

# IKE Public Key Databases and Commands

The `ikecert` command manipulates the local system's public key databases. You use this command when the `ike/config` file requires public key certificates. Because IKE 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`.

The `ikecert` command also handles key storage. Keys can be stored on disk, on an attached Sun Crypto Accelerator 6000 board, or in a softtoken keystore. The softtoken keystore is available when the metaslot in the Cryptographic Framework is used to communicate with the hardware device. The `ikecert` command uses the PKCS #11 library to locate key storage.

For more information, see the `ikecert`(1M) man page. For information about metaslot and the softtoken keystore, see the `cryptoadm`(1M) man page.

## `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 certificate requests. The certificates and certificate requests can also be stored by the Cryptographic Framework in the softtoken keystore, or on an attached Sun Crypto Accelerator 6000 board. The `ikecert` command uses the PKCS #11 library to locate certificate storage.

## `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 certificate request. The `-ks` option creates a self-signed certificate. The `-kc` option creates a certificate request. Keys are stored on the system in the `/etc/inet/secret/ike.privatekeys` directory, or on attached hardware with the `-T` option.

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 11–1** Correspondences Between `ikecert` Options and `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` | `auth_method dsa_sig` | An authentication method that is slightly slower than RSA. |
| `-t rsa-md5` and<br>`-t rsa-sha1` | `auth_method rsa_sig` | An authentication method that is slightly faster than DSA.<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>`-t rsa-sha1` | `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 certificate request with the `ikecert certlocal -kc` command, you send the output of the command to a PKI organization or to a certificate authority (CA). If your company runs its own PKI, you send the output to your PKI administrator. The PKI organization, the CA, or your PKI administrator then creates certificates. The certificates that the PKI or CA returns to you are input to the `certdb` subcommand. The certificate revocation list (CRL) that the PKI returns to you is input for the `certrldb` subcommand.

## `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 IKE With Self-Signed Public Key Certificates" on page 137. This command also accepts the certificate that you receive from a PKI or CA as input. For the procedure, see "How to Configure IKE With Certificates Signed by a CA" on page 142.

The certificates and public keys are stored on the system in the `/etc/inet/ike/publickeys` directory. The `-T` option stores the certificates, private keys, and public keys on attached hardware.

## `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 PKIs 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 a Certificate Revocation List" on page 151.

## `/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 `-T` option stores the keys on the Sun Crypto Accelerator 6000 board rather than in the `publickeys` 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 certificate signing request that you sent to the CA. You also install a certificate of the CA.

## `/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 or on supported hardware.

## /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. PKI organizations provide the CRLs for their certificates. You can use the ikecert certrldb command to populate the database.

# Glossary

**3DES** See Triple-DES.

**AES** Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces DES encryption as the government standard.

**anycast address** An IPv6 address that is assigned to a group of interfaces (typically belonging to different nodes). A packet that is sent to an anycast address is routed to the *nearest* interface having that address. The packet's route is in compliance with the routing protocol's measure of distance.

**anycast group** A group of interfaces with the same anycast IPv6 address. The Oracle Solaris implementation of IPv6 does not support the creation of anycast addresses and groups. However, Oracle Solaris IPv6 nodes can send traffic to anycast groups.

**asymmetric key cryptography** An encryption system in which the sender and receiver of a message use different keys to encrypt and decrypt the message. Asymmetric keys are used to establish a secure channel for symmetric key encryption. The Diffie-Hellman algorithm is an example of an asymmetric key protocol. Contrast with symmetric key cryptography.

**authentication header** An extension header that provides authentication and integrity, without confidentiality, to IP datagrams.

**autoconfiguration** The process where a host automatically configures its IPv6 address from the site prefix and the local MAC address.

**bidirectional tunnel** A tunnel that can transmit datagrams in both directions.

**Blowfish** A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

**broadcast address** IPv4 network addresses with the host portion of the address having all zeroes (10.50.0.0) or all one bits (10.50.255.255). A packet that is sent to a broadcast address from a machine on the local network is delivered to all machines on that network.

**CA** See certificate authority (CA).

**certificate authority (CA)** A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The CA guarantees the identity of the individual who is granted the unique certificate.

**certificate revocation list (CRL)** A list of public key certificates that have been revoked by a CA. CRLs are stored in the CRL database that is maintained through IKE.

**class** In IPQoS, a group of network flows that share similar characteristics. You define classes in the IPQoS configuration file.

**classless inter-domain routing (CIDR) address** An IPv4 address format that is not based on network classes (Class A, B, and C). CIDR addresses are 32 bits in length. They use the standard IPv4 dotted decimal notation format, with the addition of a network prefix. This prefix defines the network number and the network mask.

**datagram** See IP datagram.

**DES** Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.

**Diffie-Hellman algorithm** Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by the IKE protocol.

**diffserv model** Internet Engineering Task Force architectural standard for implementing differentiated services on IP networks. The major modules are classifier, meter, marker, scheduler, and dropper. IPQoS implements the classifier, meter, and marker modules. The diffserv model is described in RFC 2475, *An Architecture for Differentiated Services*.

**digital signature** A digital code that is attached to an electronically transmitted message that uniquely identifies the sender.

**domain of interpretation (DOI)** A DOI defines data formats, network traffic exchange types, and conventions for naming security-relevant information. Security policies, cryptographic algorithms, and cryptographic modes are examples of security-relevant information.

**DS codepoint (DSCP)** A 6-bit value that, when included in the DS field of an IP header, indicates how a packet must be forwarded.

**DSA** Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA-1 for input.

**dual stack** A TCP/IP protocol stack with both IPv4 and IPv6 at the network layer, with the rest of the stack being identical. When you enable IPv6 during an Oracle Solaris installation, the host receives the dual-stack version of TCP/IP.

**dynamic packet filter** See stateful packet filter.

**dynamic reconfiguration (DR)** A feature that allows you to reconfigure a system while the system is running, with little or no impact on ongoing operations. Not all Sun platforms from Oracle support DR. Some Sun platforms from Oracle might only support DR of certain types of hardware such as NICs.

**encapsulating security payload (ESP)** An extension header that provides integrity and confidentiality to datagrams. ESP is one of the five components of the IP Security Architecture (IPsec).

**encapsulation** The process of a header and payload being placed in the first packet, which is subsequently placed in the second packet's payload.

**filter** A set of rules that define the characteristics of a class in the IPQoS configuration file. The IPQoS system selects for processing any traffic flows that conform to the filters in its IPQoS configuration file. See packet filter.

**firewall** Any device or software that isolates an organization's private network or intranet from the Internet, thus protecting it from external intrusions. A firewall can include packet filtering, proxy servers, and NAT (network address translation).

**flow accounting** In IPQoS, the process of accumulating and recording information about traffic flows. You establish flow accounting by defining parameters for the `flowacct` module in the IPQoS configuration file.

**hash value** A number that is generated from a string of text. Hash functions are used to ensure that transmitted messages have not been tampered with. MD5 and SHA-1 are examples of one-way hash functions.

**header** See IP header.

**HMAC** Keyed hashing method for message authentication. HMAC is a secret key authentication algorithm. HMAC is used with an iterative cryptographic hash function, such as MD5 or SHA-1, in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

**hop** A measure that is used to identify the number of routers that separate two hosts. If three routers separate a source and destination, the hosts are four hops away from each other.

**host** A system that does not perform packet forwarding. Upon installation of Oracle Solaris, a system becomes a host by default, that is, the system cannot forward packets. A host typically has one physical interface, although it can have multiple interfaces.

**ICMP** Internet Control Message Protocol. Used to handle errors and exchange control messages.

**ICMP echo request packet** A packet sent to a machine on the Internet to solicit a response. Such packets are commonly known as “ping” packets.

**IKE** Internet Key Exchange. IKE automates the provision of authenticated keying material for IPsec security associations (SAs).

**Internet Protocol (IP)** The method or protocol by which data is sent from one computer to another on the Internet.

**IP** See Internet Protocol (IP), IPv4, IPv6.

**IP datagram** A packet of information that is carried over IP. An IP datagram contains a header and data. The header includes the addresses of the source and the destination of the datagram. Other fields in the header help identify and recombine the data with accompanying datagrams at the destination.

**IP header** Twenty bytes of data that uniquely identify an Internet packet. The header includes source and destination addresses for the packet. An option exists within the header to allow further bytes to be added.

**IP in IP encapsulation** The mechanism for tunneling IP packets within IP packets.

**IP link** A communication facility or medium over which nodes can communicate at the link layer. The link layer is the layer immediately below IPv4/IPv6. Examples include Ethernets (simple or bridged) or ATM networks. One or more IPv4 subnet numbers or prefixes are assigned to an IP link. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated LAN. When you use ARP, the scope of the ARP protocol is a single IP link.

**IP stack** TCP/IP is frequently referred to as a "stack." This refers to the layers (TCP, IP, and sometimes others) through which all data passes at both client and server ends of a data exchange.

**IPQoS** A software feature that provides an implementation of the diffserv model standard, plus flow accounting and 802.1 D marking for virtual LANs. Using IPQoS, you can provide different levels of network services to customers and applications, as defined in the IPQoS configuration file.

**IPsec** IP security. The security architecture that provides protection for IP datagrams.

**IPv4** Internet Protocol, version 4. IPv4 is sometimes referred to as IP. This version supports a 32-bit address space.

**IPv6** Internet Protocol, version 6. IPv6 supports a 128-bit address space.

**key management** The way in which you manage security associations (SAs).

**keystore name** The name that an administrator gives to the storage area, or keystore, on a network interface card (NIC). The keystore name is also called the token or the token ID.

**link layer** The layer immediately below IPv4/IPv6.

**link-local address** In IPv6, a designation that is used for addressing on a single link for purposes such as automatic address configuration. By default, the link-local address is created from the system's MAC address.

**load spreading** The process of distributing inbound or outbound traffic over a set of interfaces. With load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections. Two types of load spreading exists: inbound load spreading for inbound traffic and outbound load spreading for outbound traffic.

**local-use address** A unicast address that has only local routability scope (within the subnet or within a subscriber network). This address also can have a local or global uniqueness scope.

**marker** 1. A module in the diffserv architecture and IPQoS that marks the DS field of an IP packet with a value that indicates how the packet is to be forwarded. In the IPQoS implementation, the marker module is `dscpmk`.

2. A module in the IPQoS implementation that marks the virtual LAN tag of an Ethernet datagram with a user priority value. The user priority value indicates how datagrams are to be forwarded on a network with VLAN devices. This module is called `dlcosmk`.

**MD5** An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.

**message authentication code (MAC)** MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.

**meter** A module in the diffserv architecture that measures the rate of traffic flow for a particular class. The IPQoS implementation includes two meters, `tokenmt` and `tswtclmt`.

**minimal encapsulation** An optional form of IPv4 in IPv4 tunneling that can be supported by home agents, foreign agents, and mobile nodes. Minimal encapsulation has 8 or 12 bytes less of overhead than does IP in IP encapsulation.

**MTU** Maximum Transmission Unit. The size, given in octets, that can be transmitted over a link. For example, the MTU of an Ethernet is 1500 octets.

**multicast address** An IPv6 address that identifies a group of interfaces in a particular way. A packet that is sent to a multicast address is delivered to all of the interfaces in the group. The IPv6 multicast address has similar functionality to the IPv4 broadcast address.

**multihomed host** A system that has more than one physical interface and that does not perform packet forwarding. A multihomed host can run routing protocols.

**NAT** See network address translation.

**neighbor advertisement** A response to a neighbor solicitation message or the process of a node sending unsolicited neighbor advertisements to announce a link-layer address change.

**neighbor discovery** An IP mechanism that enables hosts to locate other hosts that reside on an attached link.

**neighbor solicitation** A solicitation that is sent by a node to determine the link-layer address of a neighbor. A neighbor solicitation also verifies that a neighbor is still reachable by a cached link-layer address.

**network address translation** NAT. The translation of an IP address used within one network to a different IP address known within another network. Used to limit the number of global IP addresses that are needed.

**network interface card (NIC)** Network adapter card that is an interface to a network. Some NICs can have multiple physical interfaces, such as the `igb` card.

**node** In IPv6, any system that is IPv6-enabled, whether a host or a router.

**outcome** The action to take as a result of metering traffic. The IPQoS meters have three outcomes, red, yellow, and green, which you define in the IPQoS configuration file.

**packet** A group of information that is transmitted as a unit over communications lines. Contains an IP header plus a payload.

**packet filter** A firewall function that can be configured to allow or disallow specified packets through a firewall.

**packet header** See IP header.

**payload** The data that is carried in a packet. The payload does not include the header information that is required to get the packet to its destination.

**per-hop behavior (PHB)** A priority that is assigned to a traffic class. The PHB indicates the precedence which flows of that class have in relation to other traffic classes.

**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.

PFS applies to authenticated key exchange only. See also Diffie-Hellman algorithm.

**physical interface** A system's attachment to a link. This attachment is often implemented as a device driver plus a network interface card (NIC). Some NICs can have multiple points of attachment, for example, `igb`.

**PKI** Public Key Infrastructure. A system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction.

**private address** An IP address that is not routable through the Internet. Private addresses can used by internal networks on hosts that do not require Internet connectivity. These addresses are defined in Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`) and often referred to as "1918" addresses.

**protocol stack** See IP stack.

**proxy server** A server that sits between a client application, such as a Web browser, and another server. Used to filter requests – to prevent access to certain web sites, for instance.

**public key cryptography** A cryptographic system that uses two different keys. The public key is known to everyone. The private key is known only to the recipient of the message. IKE provides public keys for IPsec.

**redirect** In a router, to inform a host of a better first-hop node to reach a particular destination.

**repair detection** The process of detecting when a NIC or the path from the NIC to some layer-3 device starts operating correctly after a failure.

**replay attack** In IPsec, an attack in which a packet is captured by an intruder. The stored packet then replaces or repeats the original at a later time. To protect against such attacks, a packet can contain a field that increments during the lifetime of the secret key that is protecting the packet.

**reverse tunnel** A tunnel that starts at the mobile node's care-of address and terminates at the home agent.

**router** A system that usually has more than one interface, runs routing protocols, and forwards packets. You can configure a system with only one interface as a router if the system is the endpoint of a PPP link.

**router advertisement** The process of routers advertising their presence together with various link and Internet parameters, either periodically or in response to a router solicitation message.

**router discovery** The process of hosts locating routers that reside on an attached link.

**router solicitation** The process of hosts requesting routers to generate router advertisements immediately, rather than at their next scheduled time.

**RSA** A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

**SA** See security association (SA).

**SADB** Security Associations Database. A table that specifies cryptographic keys and cryptographic algorithms. The keys and algorithms are used in the secure transmission of data.

**SCTP** See streams control transport protocol.

**security association (SA)** An association that specifies security properties from one host to a second host.

**security parameter index (SPI)** An integer that specifies the row in the security associations database (SADB) that a receiver should use to decrypt a received packet.

**security policy database (SPD)** Database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine whether a packet should be discarded, should be passed in the clear, or should be protected with IPsec.

**selector** The element that specifically defines the criteria to be applied to packets of a particular class in order to select that traffic from the network stream. You define selectors in the filter clause of the IPQoS configuration file.

**SHA-1** Secure Hashing Algorithm. The algorithm operates on any input length less than $2^{64}$ to produce a message digest. The SHA-1 algorithm is input to DSA.

**site-local-use address** A designation that is used for addressing on a single site.

**smurf attack** To use ICMP echo request packets directed to an IP broadcast address or multiple broadcast addresses from remote locations to create severe network congestion or outages.

**sniff** To eavesdrop on computer networks – frequently used as part of automated programs to sift information, such as clear-text passwords, off the wire.

**SPD** See security policy database (SPD).

**SPI** See security parameter index (SPI).

**spoof** To gain unauthorized access to a computer by sending a message to it with an IP address indicating that the message is coming from a trusted host. To engage in IP spoofing, a hacker must first use a variety of techniques to find an IP address of a trusted host and then modify the packet headers so that it appears that the packets are coming from that host.

**stack** See IP stack.

**standby** A physical interface that is not used to carry data traffic unless some other physical interface has failed.

**stateful packet filter** A packet filter that can monitor the state of active connections and use the information obtained to determine which network packets to allow through the firewall. By tracking and matching requests and replies, a stateful packet filter can screen for a reply that doesn't match a request.

**stateless autoconfiguration** The process of a host generating its own IPv6 addresses by combining its MAC address and an IPv6 prefix that is advertised by a local IPv6 router.

**stream control transport protocol** A transport layer protocol that provides connection-oriented communications in a manner similar to TCP. Additionally, SCTP supports multihoming, in which one of the endpoints of the connection can have more than one IP address.

**symmetric key cryptography** An encryption system in which the sender and receiver of a message share a single, common key. This common key is used to encrypt and decrypt the message. Symmetric keys are used to encrypt the bulk of data transmission in IPsec. DES is one example of a symmetric key system.

**TCP/IP** TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet).

**Triple-DES** Triple-Data Encryption Standard. A symmetric-key encryption method. Triple-DES requires a key length of 168 bits. Triple-DES is also written as 3DES.

**tunnel** The path that is followed by a datagram while it is encapsulated. See encapsulation.

**unicast address** An IPv6 address that identifies a single interface of an IPv6-enabled node. The parts of the unicast address are site prefix, subnet ID, and interface ID.

**user-priority** A 3-bit value that implements class-of-service marks, which define how Ethernet datagrams are forwarded on a network of VLAN devices.

**virtual LAN (VLAN) device** Network interfaces that provide traffic forwarding at the Ethernet (datalink) level of the IP protocol stack.

**virtual network** A combination of software and hardware network resources and functionality that are administered together as a single software entity. An *internal* virtual network consolidates network resources onto a single system, sometimes referred to as a “network in a box.”

**virtual network interface (VNIC)** A pseudo-interface that provides virtual network connectivity whether or not it is configured on a physical network interface. Containers such as exclusive IP zones are configured above VNICs to form a virtual network.

**virtual private network (VPN)** A single, secure, logical network that uses tunnels across a public network such as the Internet.

# Index

## C

## K



## L

## R



## S

## U



## V



## W



## X



## Z