

Securing the Network in Oracle® Solaris 11.2

ORACLE®

Part No: E36838-02
August 2014

Copyright © 1999, 2014, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

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.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible or and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Copyright © 1999, 2014, Oracle et/ou ses affiliés. Tous droits réservés.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l'utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique:

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.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour ce type d'applications.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée d'The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation.

Contents

Using This Documentation	13
1 Using Link Protection in Virtualized Environments	15
What's New in Network Security in Oracle Solaris 11.2	15
About Link Protection	15
Link Protection Types	16
Configuring Link Protection	17
▼ How to Enable Link Protection	18
▼ How to Disable Link Protection	18
▼ How to Specify IP Addresses to Protect Against IP Spoofing	19
▼ How to Specify DHCP Clients to Protect Against DHCP Spoofing	20
▼ How to View Link Protection Configuration and Statistics	20
2 Tuning Your Network	23
Tuning the Network	23
▼ How to Disable the Network Routing Daemon	24
▼ How to Disable Broadcast Packet Forwarding	24
▼ How to Disable Responses to Echo Requests	25
▼ How to Set Strict Multihoming	26
▼ How to Set Maximum Number of Incomplete TCP Connections	26
▼ How to Set Maximum Number of Pending TCP Connections	27
▼ How to Specify a Strong Random Number for Initial TCP Connection	28
▼ How to Prevent ICMP Redirects	28
▼ How to Reset Network Parameters to Secure Values	29
3 Web Servers and the Secure Sockets Layer Protocol	31
SSL Kernel Proxy Encrypts Web Server Communications	31
Protecting Web Servers With the SSL Kernel Proxy	33
▼ How to Configure an Apache 2.2 Web Server to Use the SSL Kernel Proxy	33

▼ How to Configure an Oracle iPlanet Web Server to Use the SSL Kernel Proxy	35
▼ How to Configure the SSL Kernel Proxy to Fall Back to the Apache 2.2 SSL	36
▼ How to Use the SSL Kernel Proxy in Zones	39
4 About IP Filter in Oracle Solaris	41
Introduction to IP Filter	41
Information Sources for Open Source IP Filter	42
IP Filter Packet Processing	42
Guidelines for Using IP Filter	44
Using IP Filter Configuration Files	45
Using IP Filter Rule Sets	45
Using IP Filter's Packet Filtering Feature	46
Using IP Filter's NAT Feature	49
Using IP Filter's Address Pools Feature	50
IPv6 for IP Filter	51
IP Filter Man Pages	52
5 Configuring IP Filter	55
Configuring the IP Filter Service	55
▼ How to Display IP Filter Service Defaults	56
▼ How to Create IP Filter Configuration Files	57
▼ How to Enable and Refresh IP Filter	58
▼ How to Disable Packet Reassembly	59
▼ How to Enable Loopback Filtering	60
▼ How to Disable Packet Filtering	61
Working With IP Filter Rule Sets	61
Managing Packet Filtering Rule Sets for IP Filter	62
Managing NAT Rules for IP Filter	69
Managing Address Pools for IP Filter	71
Displaying Statistics and Information for IP Filter	73
▼ How to View State Tables for IP Filter	73
▼ How to View State Statistics for IP Filter	74
▼ How to View IP Filter Tunable Parameters	75
▼ How to View NAT Statistics for IP Filter	75
▼ How to View Address Pool Statistics for IP Filter	76
Working With Log Files for IP Filter	76
▼ How to Set Up a Log File for IP Filter	77

▼ How to View IP Filter Log Files	78
▼ How to Flush the Packet Log Buffer	79
▼ How to Save Logged Packets to a File	79
IP Filter Configuration File Examples	80
6 About IP Security Architecture	85
Introduction to IPsec	85
IPsec Packet Flow	86
IPsec Security Associations	89
Key Management for IPsec Security Associations	89
IPsec Protection Protocols	90
Authentication Header	91
Encapsulating Security Payload	91
Authentication and Encryption Algorithms in IPsec	92
IPsec Protection Policies	93
Transport and Tunnel Modes in IPsec	93
Virtual Private Networks and IPsec	95
IPsec and FIPS 140	96
IPsec and NAT Traversal	97
IPsec and SCTP	98
IPsec and Oracle Solaris Zones	98
IPsec and Virtual Machines	98
IPsec Configuration Commands and Files	99
7 Configuring IPsec	101
Protecting Network Traffic With IPsec	101
▼ How to Secure Network Traffic Between Two Servers With IPsec	102
▼ How to Use IPsec to Protect Web Server Communication With Other Servers	106
Protecting a VPN With IPsec	108
Examples of Protecting a VPN With IPsec by Using Tunnel Mode	109
Description of the Network Topology for the IPsec Tasks to Protect a VPN	110
▼ How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode	112
Additional IPsec Tasks	116
▼ How to Manually Create IPsec Keys	117
▼ How to Configure a Role for Network Security	119
▼ How to Verify That Packets Are Protected With IPsec	123

8 About Internet Key Exchange	125
Introduction to IKE	125
IKE Concepts and Terminology	126
How IKE Works	126
Comparison of IKEv2 and IKEv1	130
IKEv2 Protocol	131
IKEv2 Configuration Choices	131
IKEv2 Policy for Public Certificates	132
IKEv1 Protocol	132
IKEv1 Key Negotiation	132
IKEv1 Configuration Choices	133
9 Configuring IKEv2	135
Configuring IKEv2	135
Configuring IKEv2 With Preshared Keys	136
▼ How to Configure IKEv2 With Preshared Keys	136
▼ How to Add a New Peer When Using Preshared Keys in IKEv2	139
Initializing the Keystore to Store Public Key Certificates for IKEv2	141
▼ How to Create and Use a Keystore for IKEv2 Public Key Certificates	142
Configuring IKEv2 With Public Key Certificates	144
▼ How to Configure IKEv2 With Self-Signed Public Key Certificates	145
▼ How to Configure IKEv2 With Certificates Signed by a CA	151
▼ How to Set a Certificate Validation Policy in IKEv2	153
▼ How to Handle Revoked Certificates in IKEv2	155
▼ How to Generate and Store Public Key Certificates for IKEv2 in Hardware	157
10 Configuring IKEv1	161
Configuring IKEv1	161
Configuring IKEv1 With Preshared Keys	162
▼ How to Configure IKEv1 With Preshared Keys	162
▼ How to Update IKEv1 for a New Peer System	165
Configuring IKEv1 With Public Key Certificates	166
▼ How to Configure IKEv1 With Self-Signed Public Key Certificates	168
▼ How to Configure IKEv1 With Certificates Signed by a CA	173
▼ How to Generate and Store Public Key Certificates for IKEv1 in Hardware	178
▼ How to Handle Revoked Certificates in IKEv1	181
Configuring IKEv1 for Mobile Systems	184

▼ How to Configure IKEv1 for Off-Site Systems	184
Configuring IKEv1 to Find Attached Hardware	191
▼ How to Configure IKEv1 to Find the Sun Crypto Accelerator 6000 Board	191
11 Troubleshooting IPsec and Its Key Management Services	193
Troubleshooting IPsec and Its Key Management Configuration	193
▼ How to Prepare IPsec and IKE Systems for Troubleshooting	194
▼ How to Troubleshoot Systems Before IPsec and IKE Are Running	195
▼ How to Troubleshoot Systems When IPsec Is Running	196
Troubleshooting IPsec and IKE Semantic Errors	200
Viewing Information About IPsec and Its Keying Services	201
Viewing IPsec and Manual Key Service Properties	201
Viewing IKE Information	202
Managing IPsec and Its Keying Services	206
Configuring and Managing IPsec and Its Keying Services	206
Managing the Running IKE Daemons	208
12 IPsec and Key Management Reference	211
IPsec Reference	211
IPsec Services, Files, and Commands	211
Security Associations Database for IPsec	216
Key Management in IPsec	216
IKEv2 Reference	216
IKEv2 Utilities and Files	217
IKEv2 Service	217
IKEv2 Daemon	218
IKEv2 Configuration File	219
ikeadm Command for IKEv2	219
IKEv2 Preshared Keys File	219
IKEv2 ikev2cert Command	220
IKEv1 Reference	220
IKEv1 Utilities and Files	220
IKEv1 Service	221
IKEv1 Daemon	222
IKEv1 Configuration File	223
IKEv1 ikeadm Command	223
IKEv1 Preshared Keys Files	224
IKEv1 Public Key Databases and Commands	224

Glossary	229
Index	237

Tables

TABLE 1-1	Configuring Link Protection Task Map	17
TABLE 2-1	Tuning Your Network Task Map	23
TABLE 5-1	Configuring the IP Filter Service Task Map	55
TABLE 5-2	Working With IP Filter Rule Sets Task Map	61
TABLE 5-3	Displaying IP Filter Statistics and Information Task Map	73
TABLE 5-4	Working With IP Filter Log Files Task Map	76
TABLE 6-1	Protections Provided by AH and ESP in IPsec	92
TABLE 6-2	Selected IPsec Configuration Commands and Files	99
TABLE 7-1	Protecting Network Traffic With IPsec Task Map	102
TABLE 7-2	Additional IPsec Tasks Task Map	116
TABLE 8-1	IKEv2 and IKEv1 Implementation in Oracle Solaris	130
TABLE 9-1	Configuring IKEv2 With Public Key Certificates Task Map	144
TABLE 10-1	Configuring IKEv1 With Public Key Certificates Task Map	167
TABLE 10-2	Configuring IKEv1 for Mobile Systems Task Map	184
TABLE 12-1	IKEv2 Service Name, Commands, Configuration and Key Storage Locations, and Hardware Devices	217
TABLE 12-2	IKEv1 Service Name, Commands, Configuration and Key Storage Locations, and Hardware Devices	221
TABLE 12-3	Correspondences Between <code>ikecert</code> Options and <code>ike/config</code> Entries in IKEv1	225

Examples

EXAMPLE 3-1	Configuring an Apache 2.2 Web Server to Use the SSL Kernel Proxy	39
EXAMPLE 5-1	Activating a Different Packet Filtering Rule Set	64
EXAMPLE 5-2	Reloading an Updated Packet Filtering Rule Set	64
EXAMPLE 5-3	Removing a Packet Filtering Rule Set	65
EXAMPLE 5-4	Appending Rules to the Active Packet Filtering Rule Set	66
EXAMPLE 5-5	Appending Rules to the Inactive Rule Set	66
EXAMPLE 5-6	Switching Between the Active and Inactive Packet Filtering Rule Sets	67
EXAMPLE 5-7	Removing an Inactive Packet Filtering Rule Set From the Kernel	68
EXAMPLE 5-8	Removing NAT Rules	69
EXAMPLE 5-9	Appending Rules to the NAT Rule Set	70
EXAMPLE 5-10	Removing an Address Pool	71
EXAMPLE 5-11	Appending Rules to an Address Pool	73
EXAMPLE 5-12	Viewing State Tables for IP Filter	74
EXAMPLE 5-13	Viewing State Statistics for IP Filter	74
EXAMPLE 5-14	Viewing NAT Statistics for IP Filter	75
EXAMPLE 5-15	Viewing Address Pool Statistics for IP Filter	76
EXAMPLE 5-16	Creating an IP Filter Log	77
EXAMPLE 5-17	Viewing IP Filter Log Files	79
EXAMPLE 5-18	Flushing the Packet Log Buffer	79
EXAMPLE 5-19	Saving Logged Packets to a File	80
EXAMPLE 5-20	IP Filter Host Configuration	81
EXAMPLE 5-21	IP Filter Server Configuration	81
EXAMPLE 5-22	IP Filter Router Configuration	83
EXAMPLE 7-1	Configuring IPsec Policy Remotely by Using an ssh Connection	105
EXAMPLE 7-2	Configuring IPsec Policy to Run in FIPS 140 Mode	106
EXAMPLE 7-3	Creating a Tunnel That All Subnets Can Use	109
EXAMPLE 7-4	Creating a Tunnel That Connects Two Subnets Only	110
EXAMPLE 7-5	Creating and Assigning a Network Management and Security Role	120
EXAMPLE 7-6	Dividing Network Security Responsibilities Between Roles	121
EXAMPLE 7-7	Enabling a Trusted User to Configure and Manage IPsec	121

EXAMPLE 9-1	Using Different Local and Remote IKEv2 Preshared Keys	138
EXAMPLE 9-2	Creating a Self-Signed Certificate With a Limited Lifetime	150
EXAMPLE 9-3	Verifying a Public Key Certificate by Its Fingerprint	150
EXAMPLE 9-4	Changing the Time That a System Waits For IKEv2 Certificate Verification	156
EXAMPLE 10-1	Refreshing an IKEv1 Preshared Key	164
EXAMPLE 10-2	Using rsa_encrypt When Configuring IKEv1	176
EXAMPLE 10-3	Pasting a CRL Into the Local cert1db Database for IKEv1	183
EXAMPLE 10-4	Configuring a Central Computer That Uses IKEv1 to Accept Protected Traffic From a Mobile System	187
EXAMPLE 10-5	Configuring a System Behind a NAT With IPsec and IKEv1	188
EXAMPLE 10-6	Accepting Self-Signed Certificates From a Mobile System	189
EXAMPLE 10-7	Using Self-Signed Certificates to Contact a Central System	190
EXAMPLE 10-8	Finding and Using Metaslot Tokens	192
EXAMPLE 11-1	Fixing an Invalid IKEv2 Configuration	198
EXAMPLE 11-2	Fixing a No Matching Rule Message	199
EXAMPLE 11-3	Setting a New Debug Level on a Running IKE Daemon	199

Using This Documentation

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

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at <http://www.oracle.com/pls/topic/lookup?ctx=E36784>.

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.

Feedback

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

◆◆◆ CHAPTER 1

Using Link Protection in Virtualized Environments

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

- “What's New in Network Security in Oracle Solaris 11.2” on page 15
- “About Link Protection” on page 15
- “Configuring Link Protection” on page 17

What's New in Network Security in Oracle Solaris 11.2

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

IKE Version 2 (IKEv2) provides automatic key management for IPsec using the latest version of the IKE protocol. IKEv2 and IPsec use cryptographic algorithms from the Cryptographic Framework feature of Oracle Solaris.

Note - The Cryptographic Framework feature of Oracle Solaris is validated for FIPS 140-2, Level 1. For IKE's use of FIPS 140 mode, see [Table 8-1](#). For hardware and software details, see [Oracle FIPS 140 Software Validations \(http://www.oracle.com/technetwork/topics/security/fips140-software-validations-1703049.html\)](http://www.oracle.com/technetwork/topics/security/fips140-software-validations-1703049.html).

IKE Version 1 (IKEv1) support is still available. For more information, see [Chapter 8, “About Internet Key Exchange”](#).

About Link Protection

With the increasing adoption of virtualization in system configurations, guest virtual machines (VMs) can be given exclusive access to a physical or virtual link by the host administrator. This

configuration improves network performance by allowing the virtual environment's network traffic to be isolated from the wider traffic that is received or sent by the host system. At the same time, this configuration can expose the system and the entire network to the risk of harmful packets that a guest environment might generate.

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

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

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

Link Protection Types

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

`mac-nospoof`

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

`ip-nospoof`

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

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

`dhcp-nospoof`

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

You can add allowed clients by using the `allowed-dhcp-cids` link property. Entries in the `allowed-dhcp-cids` list must be formatted as specified in the [dhcpage\(1M\)](#) 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 20](#).

For fuller descriptions of these protection types, see the [`dladm\(1M\)`](#) man page.

Configuring Link Protection

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

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

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

TABLE 1-1 Configuring Link Protection Task Map

Task	Description	For Instructions
Enable link protection.	Restricts the packets that are sent from a link and protects links from spoofing.	“How to Enable Link Protection” on page 18
Disable link protection.	Removes link protections.	“How to Disable Link Protection” on page 18
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 19
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 20
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 20

▼ How to Enable Link Protection

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

Before You Begin You must become an administrator who is assigned the Network Link Security rights profile. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

1. View the available link protection types.

```
# dladm show-linkprop -p protection
LINK    PROPERTY    PERM VALUE    EFFECTIVE    DEFAULT    POSSIBLE
net0    protection  rw --         --         --         mac-nospoof,
                                             restricted,
                                             ip-nospoof,
                                             dhcp-nospoof
```

For a description of the possible types, see [“Link Protection Types”](#) on page 16 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
```



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

3. Verify that the link protections are enabled.

```
# dladm show-linkprop -p protection vnic0
LINK    PROPERTY    PERM VALUE    EFFECTIVE    DEFAULT    POSSIBLE
net0    protection  rw  mac-nospoof  mac-nospoof  --         mac-nospoof,
                                             restricted,
                                             ip-nospoof  ip-nospoof  --         ip-nospoof,
                                             dhcp-nospoof  dhcp-nospoof  --         dhcp-nospoof
```

▼ How to Disable Link Protection

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

Before You Begin You must become an administrator who is assigned the Network Link Security rights profile. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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

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

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

```
# dladm show-linkprop -p protection vnic0
LINK    PROPERTY    PERM VALUE    EFFECTIVE    DEFAULT    POSSIBLE
net0    protection  rw -- -- -- mac-nospoof,
restricted,
ip-nospoof,
dhcp-nospoof
```

▼ How to Specify IP Addresses to Protect Against IP Spoofing

Before You Begin The ip-nospoof protection type is enabled, as shown in [“How to Enable Link Protection”](#) on page 18.

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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

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

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

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

The following example shows how to add the IP addresses 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 18](#).

You must become an administrator who is assigned the Network Link Security rights profile. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

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

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

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

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

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

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

For more information, see the [`dladm\(1M\)`](#) 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 [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

1. View the link protection property values.

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

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

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

```

                                restricted  restricted      restricted,
                                ip-nospoof ip-nospoof     ip-nospoof,
                                dhcp-nospoof dhcp-nospoof   dhcp-nospoof
vnic0  allowed-ips             rw  10.0.0.1,    10.0.0.1,  --  --
                                10.0.0.2    10.0.0.2
vnic0  allowed-dhcp-cids      rw  hello       hello     --  --

```

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

2. View the link protection statistics.

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

```

# dlstat -A
...
vnic0
  mac_misc_stat
    multircv                0
    brdcstrcv               0
    multixmt                 0
    brdcstxmt               0
    multircvbytes           0
    bcstrcvbytes            0
    multixmtbytes           0
    bcstxmtbytes            0
    txerrors                 0
    macspoofer               0 <-----
    ipspoofer                0 <-----
    dhcspoofer               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 `dhcspoofer` statistics:

```

# kstat vnic0:0:link:dhcspoofer
module: vnic0                instance: 0
name:   link                  class:   vnic
       dhcspoofer             0

```

For more information, see the [dlstat\(1M\)](#) and [kstat\(1M\)](#) man pages.

◆ ◆ ◆ **2** C H A P T E R 2

Tuning Your Network

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

Tuning the Network

TABLE 2-1 Tuning Your Network Task Map

Task	Description	For Instructions
Disable the network routing daemon.	Limits access to systems by would-be network sniffers.	“How to Disable the Network Routing Daemon” on page 24
Prevent the dissemination of information about the network topology.	Prevents the broadcast of packets.	“How to Disable Broadcast Packet Forwarding” on page 24
	Prevents responses to broadcast echo requests and multicast echo requests.	“How to Disable Responses to Echo Requests” on page 25
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 26
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 26
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 27
Verify that strong random numbers are generated 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 28
Prevent ICMP redirection.	Removes indicators of the network topology.	“How to Prevent ICMP Redirects” on page 28
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 29

▼ 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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, 2014 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, 2014 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 11.2 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 “[Using Your Assigned Administrative Rights](#)” in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

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 11.2 Tunable Parameters Reference Manual](#)” and the [ipadm\(1M\)](#) man page.

For more information about the use of strict multihoming, see “[How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode](#)” on page 112.

▼ How to Set Maximum Number of Incomplete TCP Connections

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

Before You Begin You must become an administrator who is assigned the Network Management rights profile. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 11.2 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 11.2 Tunable Parameters Reference Manual”](#) and the [ipadm\(1M\)](#) man page.

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

This procedure ensures that the TCP initial sequence number generation parameter complies with [RFC 6528 \(http://www.ietf.org/rfc/rfc6528.txt\)](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 “[Using Your Assigned Administrative Rights](#)” in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

1. **Verify that the default value for the `TCP_STRONG_ISS` variable is 2.**

```
# grep TCP_STRONG /etc/default/inetinit
# TCP_STRONG_ISS sets the TCP initial sequence number generation parameters.
# Set TCP_STRONG_ISS to be:
TCP_STRONG_ISS=2
```

2. **If the value of `TCP_STRONG_ISS` is not 2, change it to 2.**

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

3. **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 “[Using Your Assigned Administrative Rights](#)” in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

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

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

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

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

2. Prevent sending ICMP redirect messages.

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

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

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

For more information, see “[send_redirects \(ipv4 or ipv6\)](#)” in “[Oracle Solaris 11.2 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 “[Using Your Assigned Administrative Rights](#)” in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

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 11.2 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 11.2 Tunable Parameters Reference Manual](#)”.

See Also [ipadm\(1M\)](#) man page

Web Servers and the Secure Sockets Layer Protocol

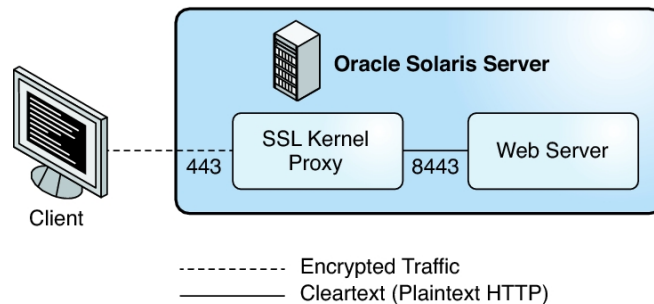
This chapter explains how to use the Secure Sockets Layer (SSL) protocol to encrypt and accelerate web server communications on your Oracle Solaris system.

- [“SSL Kernel Proxy Encrypts Web Server Communications” on page 31](#)
- [“Protecting Web Servers With the SSL Kernel Proxy” on page 33](#)

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.

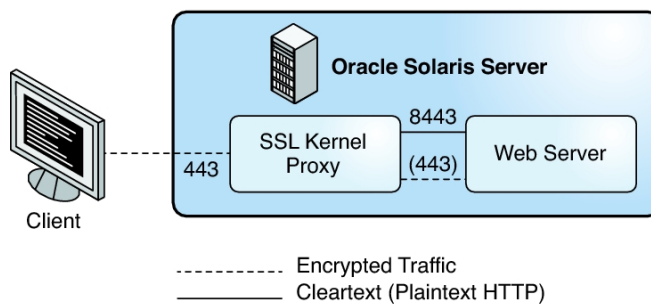
Figure 3-1 shows a basic scenario with a web server that is using the SSL kernel proxy. The SSL kernel proxy is configured on port 443, whereas the web server is configured on port 8443, where it receives unencrypted HTTP communications.

- 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

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 33](#)
- [“How to Configure an Oracle iPlanet Web Server to Use the SSL Kernel Proxy” on page 35](#)
- [“How to Configure the SSL Kernel Proxy to Fall Back to the Apache 2.2 SSL” on page 36](#)
- [“How to Use the SSL Kernel Proxy in Zones” on page 39](#)

▼ 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 PKCS11-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 \(http://www.oracle.com/technetwork/middleware/ipplanetwebserver-098726.html?ssSourceSiteId=ocomen\)](http://www.oracle.com/technetwork/middleware/ipplanetwebserver-098726.html?ssSourceSiteId=ocomen). For instructions, see [Oracle iPLANET WEB SERVER 7.0.15 \(http://docs.oracle.com/cd/E18958_01/index.htm\)](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 “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”.

1. Stop the web server.

Use the administrator web interface to stop the server. For instructions, see [Oracle iPLANET WEB SERVER 7.0.15 \(http://docs.oracle.com/cd/E18958_01/index.htm\)](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 33.

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 PKCS11-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 \(http://docs.oracle.com/cd/E18958_01/index.htm\)](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. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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
```

Note - For FIPS 140 compliance, the minimum length RSA key is 2048. For more information, see [“Using a FIPS 140 Enabled System in Oracle Solaris 11.2”](#).

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 any space characters.

- 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 unencrypted communications.**

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. You can start the web

server in the non-global zone after you configure the service instance with 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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
```


About IP Filter in Oracle Solaris

This chapter provides an overview of the IP Filter feature of Oracle Solaris. For IP Filter tasks, see [Chapter 5, “Configuring IP Filter”](#).

This chapter contains the following information:

- [“Introduction to IP Filter” on page 41](#)
- [“IP Filter Packet Processing” on page 42](#)
- [“Guidelines for Using IP Filter” on page 44](#)
- [“Using IP Filter Configuration Files” on page 45](#)
- [“Using IP Filter Rule Sets” on page 45](#)
- [“IPv6 for IP Filter” on page 51](#)
- [“IP Filter Man Pages” on page 52](#)

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.

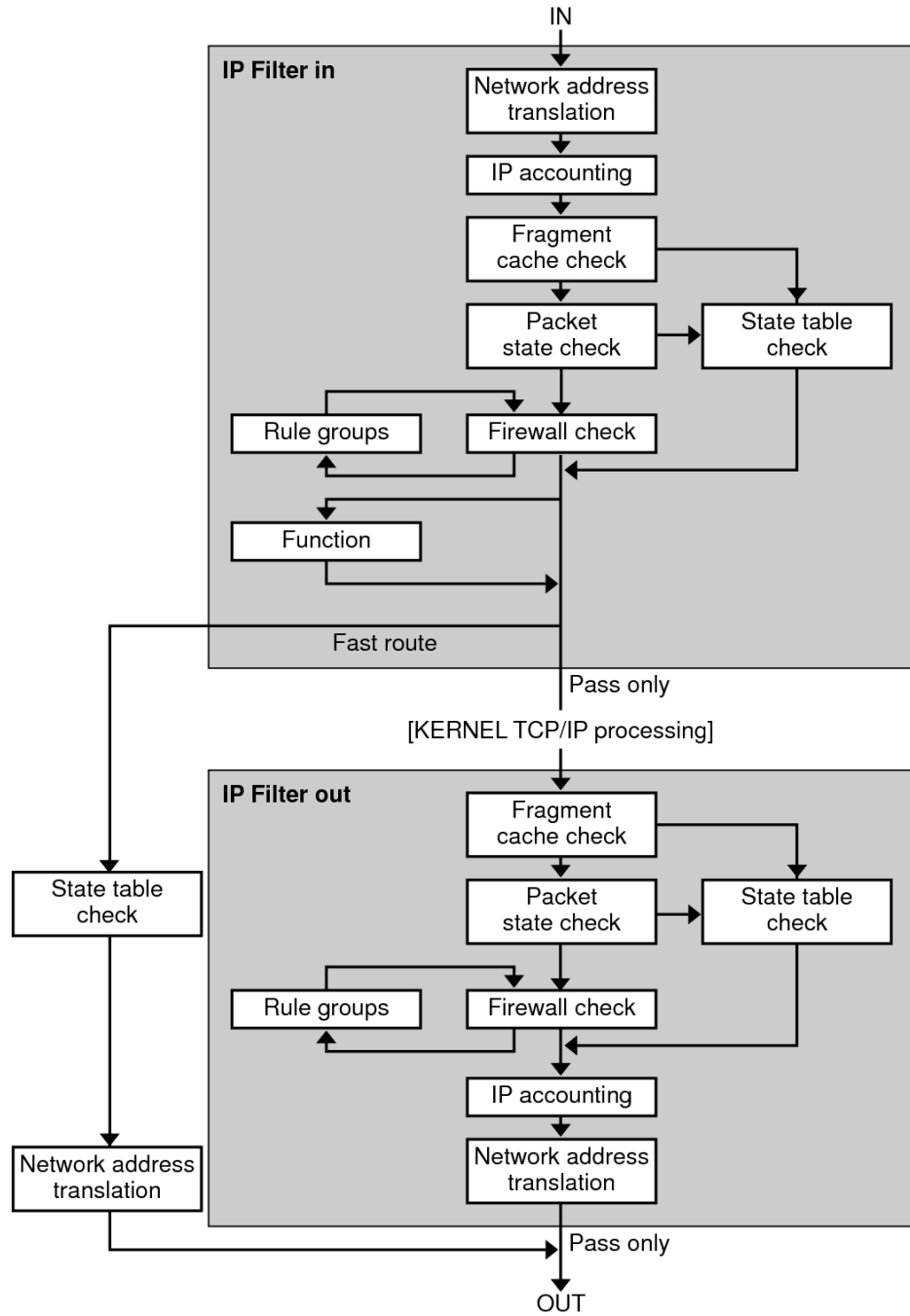
Information Sources for Open Source IP Filter

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, “Introduction to the Service Management Facility,”](#) in [“Managing System Services in Oracle Solaris 11.2”](#). For information on the step-by-step

procedures that are associated with SMF, see [Chapter 3, “Administering Services,”](#) in [“Managing System Services in Oracle Solaris 11.2”](#).

- 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 the IP Filter Service”](#) on page 55.
- To administer IP Filter, you must assume the root role or be assigned the IP Filter Management rights profile. You can assign the IP Filter Management rights profile to a user or to a role that you create. To create the role and assign the role to a user, see [“Creating a Role”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).
- 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 80. 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 can accelerate 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.

<code>block</code>	Prevents the packet from passing through the filter.
<code>pass</code>	Allows the packet through the filter.
<code>log</code>	Logs the packet but does not determine if the packet is blocked or passed. Use the <code>ipmon</code> command to view the log.

count	Includes the packet in the filter statistics. Use the <code>ipfstat</code> command to view the statistics.
skip <i>number</i>	Makes the filter skip over <i>number</i> 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.

- 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.
- 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 <code>ipmon</code> command to view the log.
quick	Executes the rule containing the <code>quick</code> option if there is a packet match. All further rule checking stops.
on <i>interface-name</i>	Applies the rule only if the packet is moving in or out of the specified interface.
dup- to <i>interface-name</i>	Copies the packet and sends the duplicate out on <i>interface-name</i> to an optionally specified IP address.

Note - The `dup- to` option in a rule allows the network administrator to create a *network tap*. Although this option is still supported in Oracle Solaris, its importance is largely diminished. Modern switches are available that enable you to directly configure their ports to perform network taps, which voids the need to define this function in a rule. Refer to your switch documentation for configuring ports to tap the network.

to <i>interface-name</i>	Moves the packet to an outbound queue on <i>interface-name</i> .
--------------------------	--

- 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.

<code>tos</code>	Filters the packet based on the type-of-service value expressed as either a hexadecimal or a decimal integer.
<code>ttl</code>	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.
<code>proto</code>	Matches a specific protocol. You can use any of the protocol names specified in the <code>/etc/protocols</code> file, or use a decimal number to represent the protocol. The keyword <code>tcp/udp</code> can be used to match either a TCP or a UDP packet.
<code>from/to/all/any</code>	Matches any or all of the following: the source IP address, the destination IP address, and the port number. The <code>all</code> keyword is used to accept packets from all sources and to all destinations.
<code>with</code>	Matches specified attributes associated with the packet. Insert either the word <code>not</code> or the word <code>no</code> in front of the keyword in order to match the packet only if the option is not present.
<code>flags</code>	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.
<code>icmp-type</code>	Filters according to ICMP type. This keyword is used only when the <code>proto</code> option is set to <code>icmp</code> and is not used if the <code>flags</code> option is used.
<code>keep keep-options</code>	Determines the information that is kept for a packet. The <i>keep-options</i> that are available include the <code>state</code> option. The <code>state</code> option keeps information about the session and can be kept for TCP, UDP, and ICMP packets.
<code>head number</code>	Creates a new group for filtering rules, which is denoted by the number <i>number</i> .
<code>group number</code>	Adds the rule to group number <i>number</i> 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 62](#). For an explanation of the IP address scheme (192.168.0.0/16) shown in the example, see [Chapter 1, “Planning For Network Deployment,” in “Planning for Network Deployment in Oracle Solaris 11.2”](#).

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 must create 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:

<code>map</code>	Maps one IP address or network to another IP address or network in an unregulated round-robin process.
<code>rdr</code>	Redirects packets from one IP address and port pair to another IP address and port pair.
<code>bimap</code>	Establishes a bidirectional NAT between an external IP address and an internal IP address.
<code>map-block</code>	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:

<code>ipmask</code>	Designates the network mask.
<code>dstipmask</code>	Designates the address that <code>ipmask</code> is translated to.
<code>mapport</code>	Designates <code>tcp</code> , <code>udp</code> , or <code>tcp/udp</code> 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 for a group of address/netmask pairs. Address pools 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, then 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
```

<code>table</code>	Defines the reference for the multiple addresses.
--------------------	---

role	Specifies the role of the pool in IP Filter. The only role you can reference is <code>ipf</code> .
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.

Note - For more information on jumbograms, see [IPv6 Jumbograms, RFC 2675 \(http://www.ietf.org/rfc/rfc2675.txt\)](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 tasks associated with IP Filter, see [Chapter 5, “Configuring IP Filter”](#).

IP Filter Man Pages

The following man pages cover IP Filter.

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.

Configuring IP Filter

This chapter provides step-by-step instructions for IP Filter tasks. For overview information, see [Chapter 4, “About IP Filter in Oracle Solaris”](#).

This chapter covers the following topics:

- [“Configuring the IP Filter Service” on page 55](#)
- [“Working With IP Filter Rule Sets” on page 61](#)
- [“Displaying Statistics and Information for IP Filter” on page 73](#)
- [“Working With Log Files for IP Filter” on page 76](#)
- [“IP Filter Configuration File Examples” on page 80](#)

Configuring the IP Filter Service

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

TABLE 5-1 Configuring the IP Filter Service 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 56
Customize packet filtering rule sets for network traffic, packets over a NAT, and address pools.	“How to Create IP Filter Configuration Files” on page 57
Enable, refresh, or disable the IP Filter service.	“How to Enable and Refresh IP Filter” on page 58
Modify the default setting for packets that arrive in fragments.	“How to Disable Packet Reassembly” on page 59
Filter traffic between zones on your system.	“How to Enable Loopback Filtering” on page 60
Stop using IP Filter.	“How to Disable Packet Filtering” on page 61

▼ 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 default file locations. These files do not exist until you create them. If you change the location of a configuration file, you must change the property value for that file. For the procedure, see [“How to Create IP Filter Configuration Files”](#) on page 57.

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 57.

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 57.

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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 46. For examples of configuration files, see [“IP Filter Configuration File Examples”](#) on page 80, 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 the default file name, `/etc/ipf/ipnat.conf`. If you use a different name, you must 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 49](#).

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 the default file name, `/etc/ipf/ippool.conf`. If you use a different name, you must 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 50](#).

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 60](#). 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 59](#).

▼ 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 [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

You have completed [“How to Create IP Filter Configuration Files” on page 57](#).

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 61.

▼ 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 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 the value of `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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 the value of `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 58](#).

Before You Begin You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 61](#).

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 lists 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 62
View an inactive packet filtering rule set.	“How to View the Inactive Packet Filtering Rule Set” on page 63
Activate a different active rule set.	“How to Activate a Different or Updated Packet Filtering Rule Set” on page 63
Remove a rule set.	“How to Remove a Packet Filtering Rule Set” on page 64

Task	For Instructions
Add rules to the rule sets.	“How to Append Rules to the Active Packet Filtering Rule Set” on page 65 “How to Append Rules to the Inactive Packet Filtering Rule Set” on page 66
Move between active and inactive rule sets.	“How to Switch Between Active and Inactive Packet Filtering Rule Sets” on page 67
Delete an inactive rule set from the kernel.	“How to Remove an Inactive Packet Filtering Rule Set From the Kernel” on page 68
View active NAT rules.	“How to View Active NAT Rules in IP Filter” on page 69
Remove NAT rules.	“How to Deactivate NAT Rules in IP Filter” on page 69
Add rules to active NAT rules.	“How to Append Rules to the NAT Packet Filtering Rules” on page 70
View active address pools.	“How to View Active Address Pools” on page 71
Remove an address pool.	“How to Remove an Address Pool” on page 71
Add rules to an address pool.	“How to Append Rules to an Address Pool” on page 72

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 56](#).

▼ 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 [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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.

Optionally, list the active ruleset.

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

Then, edit the `/etc/ipf/myorg.ipf.conf` configuration file, refresh the service, and list the active ruleset again.

```
# svcadm refresh network/ipfilter
# ipfstat -io
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any
block in quick on net1 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 troubleshooting. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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:
 1. Create a rule set in a file of your choice.
 2. 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 troubleshooting. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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 troubleshooting. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

● 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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

The following procedures 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 troubleshooting. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

● 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:
 1. Create additional NAT rules in a file of your choice.
 2. 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

The following procedures 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 troubleshooting. 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

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:
 1. Create additional address pools in a file of your choice.
 2. 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 65.

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 73
View statistics about packet state.	“How to View State Statistics for IP Filter” on page 74
List IP Filter tunables.	“How to View IP Filter Tunable Parameters” on page 75
View NAT statistics.	“How to View NAT Statistics for IP Filter” on page 75
View address pool statistics.	“How to View Address Pool Statistics for IP Filter” on page 76

▼ 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 77
View state, NAT, and normal log files.	“How to View IP Filter Log Files” on page 78
Flush the packet log buffer.	“How to Flush the Packet Log Buffer” on page 79
Save logged packets to a file for later reference.	“How to Save Logged Packets to a File” on page 79

▼ How to Set Up a Log File for IP Filter

By default, all log information for IP Filter is recorded by `syslog`. 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 `syslog` log file.

Before You Begin You must assume the root role.

1. Determine which `system-log` service instance is enabled.

```
% 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 enabled.

Example 5-16 Creating an IP Filter Log

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

Edit the `syslog.conf`.

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

Then, on the command line, create the file and restart the service.

```
# 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 77.

You must become an administrator who is assigned the IP Filter Management rights profile. For more information, see “[Using Your Assigned Administrative Rights](#)” in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

- **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. By issuing the `ipmon -a` command, you open another copy of `ipmon`. Then, both copies read the same log information, but 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 [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

- **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 the log file is empty, 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 troubleshooting, 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. A backslash (\) at the end of a line indicates that the rule continues on the next line.

For more detailed syntax information, see [“Configuring Packet Filtering Rules” on page 46](#).

EXAMPLE 5-20 IP Filter Host Configuration

This example shows a configuration on a host system 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 system. 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 system that acts as a web server. This system 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 ***

# 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
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 are not 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

# 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

# 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
```

About IP Security Architecture

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

This chapter covers the following topics:

- [“Introduction to IPsec” on page 85](#)
- [“IPsec Packet Flow” on page 86](#)
- [“IPsec Security Associations” on page 89](#)
- [“IPsec Protection Protocols” on page 90](#)
- [“IPsec Protection Policies” on page 93](#)
- [“Transport and Tunnel Modes in IPsec” on page 93](#)
- [“Virtual Private Networks and IPsec” on page 95](#)
- [“Virtual Private Networks and IPsec” on page 95](#)
- [“IPsec and FIPS 140” on page 96](#)
- [“IPsec and SCTP” on page 98](#)
- [“IPsec Configuration Commands and Files” on page 99](#)

To implement IPsec on your network, see [Chapter 7, “Configuring IPsec”](#). For reference information, see [Chapter 12, “IPsec and Key Management Reference”](#).

Introduction to IPsec

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

IPsec uses the following terms:

- **Security protocols** – The protection that is applied to an IP packet. The [authentication header \(AH\)](#) protects a IP packet by adding an integrity check vector (ICV) which is a hash

of the complete packet including the IP headers. The receiver is assured that the packet has not been modified. It does not provide confidentiality with encryption.

The [encapsulating security payload \(ESP\)](#) protects the payload of an IP packet. The payload of a packet can be encrypted to provide confidentiality and can ensure data integrity by using an ICV.

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

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

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

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

IPsec Packet Flow

[Figure 6-1](#) shows how an [IP packet](#) proceeds when IPsec has been invoked on an outbound packet. The flow diagram illustrates where authentication header (AH) and encapsulating security payload (ESP) entities can be applied to the packet. Subsequent sections describe how to apply these entities, as well as how to choose the algorithms.

[Figure 6-2](#) shows the IPsec inbound process.

FIGURE 6-1 IPsec Applied to Outbound Packet Process

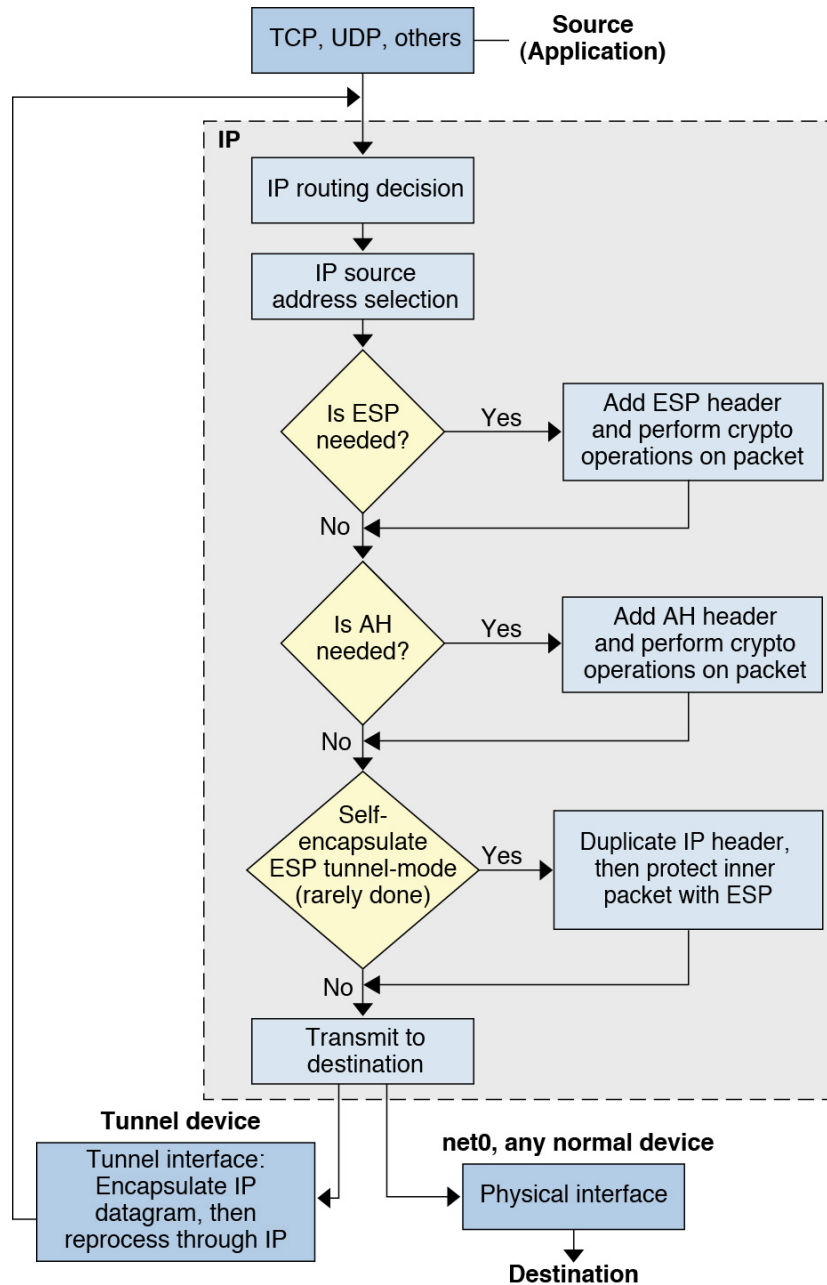
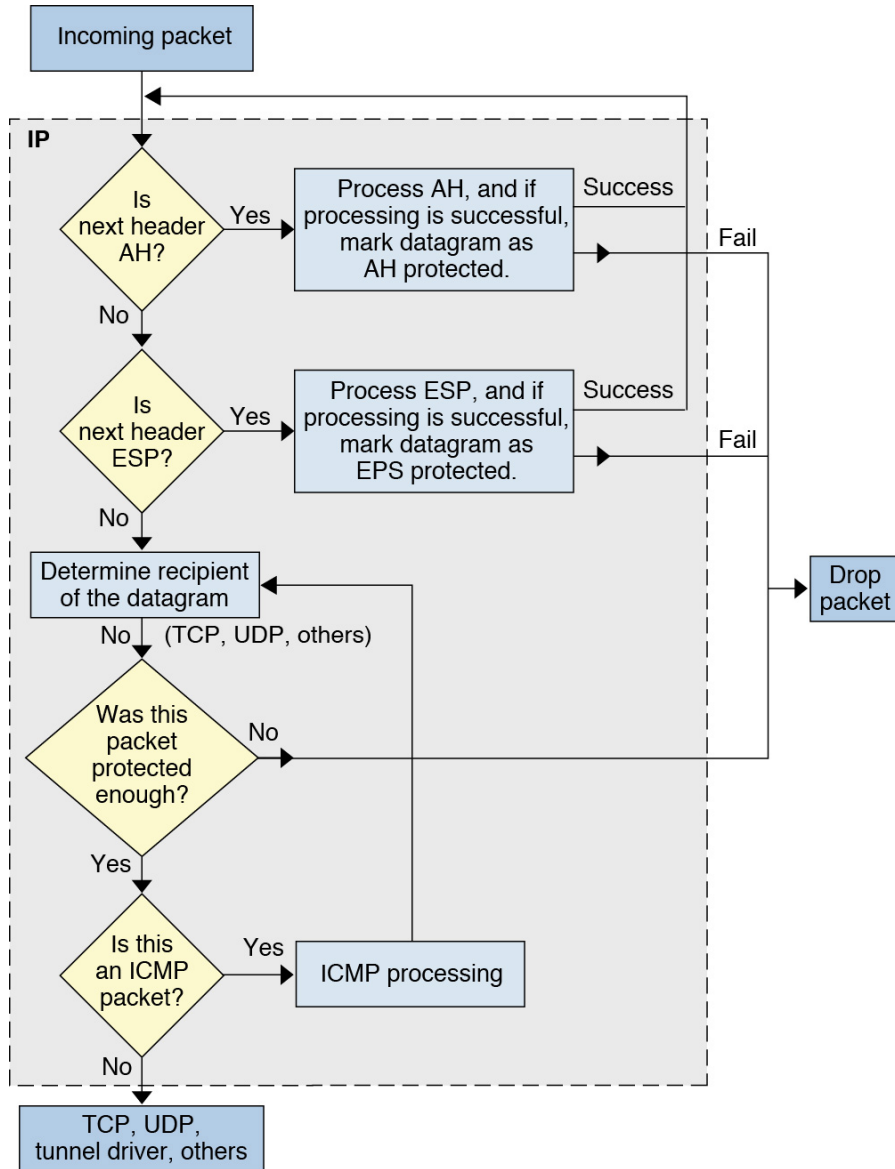


FIGURE 6-2 IPsec Applied to Inbound Packet Process



IPsec Security Associations

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

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

- The security protocol (AH or ESP)
- The destination IP address
- The [security parameter index \(SPI\)](#)

The SPI of the SA provides additional protection and is transmitted in the AH or ESP header of an IPsec-protected packet. The [ipsecah\(7P\)](#) 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 programmatically. For example, the IKE daemon and the `ipseckey` command use the PF_KEY socket interface.

For a more complete description of the IPsec SADB, see “[Security Associations Database for IPsec](#)” on page 216.

For more information about how to manage the SADB, see the [pf_key\(7P\)](#) and [ipseckey\(1M\)](#) man pages.

Key Management for IPsec Security Associations

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

IKE for IPsec SA Generation

The Internet Key Exchange (IKE) protocol handles key management automatically. Oracle Solaris 11.2 supports IKE version 2 (IKEv2) and IKE version 1 (IKEv1) of the IKE protocol.

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

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

For more information, see [“How IKE Works” on page 126](#).

To configure IKE, see [Chapter 9, “Configuring IKEv2”](#). If you are communicating with a system that does not support the IKEv2 protocol, follow the instructions in [Chapter 10, “Configuring IKEv1”](#).

Manual Keys for IPsec SA Generation

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

While the `ipseckey` command has only a limited number of general options, the command supports a rich command language. You can specify that requests be delivered by means of a programmatic interface specific for manual keying. For additional information, see the [ipseckey\(1M\)](#) and [pf_key\(7P\)](#) man pages.

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

IPsec Protection Protocols

IPsec provides two security protocols for protecting data:

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

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

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

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

Authentication Header

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



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

Encapsulating Security Payload

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

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



Encrypted

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

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

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

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

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

For further discussion, see [“Transport and Tunnel Modes in IPsec”](#) on page 93.

Security Considerations When Using AH and ESP

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

TABLE 6-1 Protections Provided by AH and ESP in IPsec

Protocol	Packet Coverage	Protection	Against Attacks
AH	Protects packet from the IP header to the end of the transport data	Provides strong integrity, data authentication: <ul style="list-style-type: none"> ■ Ensures that the receiver receives exactly what the sender sent ■ Is susceptible to replay attacks when an AH does not enable replay protection 	Replay, cut-and-paste
ESP	Protects packet from the ESP header to the end of the transport data	With encryption option, encrypts the IP payload. Ensures confidentiality	Eavesdropping
		With authentication option, provides the same payload protection as AH	Replay, cut-and-paste
		With both options, provides strong integrity, data authentication, and confidentiality	Replay, cut-and-paste, eavesdropping

Authentication and Encryption Algorithms in IPsec

IPsec security uses two types of algorithms, authentication and encryption. The AH protocol uses authentication algorithms. The ESP protocol can use encryption as well as authentication algorithms. You can obtain a list of the algorithms on your system and their properties by using the `ipsecalgs` command. For more information, see the [ipsecalgs\(1M\)](#) 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 perform encryption and authentication. The Cryptographic Framework enables IPsec to take advantage of hardware acceleration where the hardware supports it.

For more information, see the following:

- [Chapter 1, “Cryptographic Framework,”](#) in [“Managing Encryption and Certificates in Oracle Solaris 11.2 ”](#)
- [Chapter 8, “Introduction to the Oracle Solaris Cryptographic Framework,”](#) in [“Developer’s Guide to Oracle Solaris 11 Security ”](#)

IPsec Protection Policies

IPsec protection policies can be applied at the following levels:

- System-wide level
- Per-socket level

IPsec applies the system-wide policy to outbound packets and inbound packets that match an IPsec policy rule. The rule can specify a particular algorithm or allow a one of several algorithms. You can apply additional rules to outbound packets because of the additional data that is known by the system.

Inbound packets are either accepted or dropped. The decision to drop or accept an inbound packet is based on several criteria. If the criteria overlap or conflict, the rule that is parsed first is used.

You can specify exceptions to an IPsec policy that otherwise applies to most packets. That is, you can *bypass* an IPsec policy. The bypass can be system-wide or per-socket.

For traffic within a system including zones on a shared-IP address, 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. For exclusive-IP zones, policy is enforced and actual security mechanisms are 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 and [Chapter 7, “Configuring IPsec”](#).

Transport and Tunnel Modes in IPsec

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

The packets can be protected by AH, ESP, or both in each mode. The modes differ in policy application, as follows:

- In transport mode, the IP addresses in the outer header are used to determine the IPsec policy that will be applied to the packet.
- In tunnel mode, two IP headers are sent. The inner IP packet determines the IPsec policy that protects its contents.

Tunnel mode can be applied to any mix of end systems and intermediate systems, such as security gateways.

In transport mode, the IP header, the next header, and any ports that the next header supports can be used to determine 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.

Tunnel mode works only for IP-in-IP packets. In tunnel mode, IPsec policy is enforced on the contents of the inner IP packet. Different 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 packet.

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 108](#).

In Oracle Solaris, tunnel mode can be enforced only on an IP tunneling network interface. For information about tunneling interfaces, see [Chapter 4, “About IP Tunnel Administration,” in “Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2”](#). IPsec policy provides a `tunnel` keyword to select an IP tunneling network interface. When the `tunnel` keyword is present in a rule, all selectors that are specified in that rule apply to the inner packet.

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



Encrypted

In tunnel mode, the entire packet is *inside* the ESP header. The packet in [Figure 6-3](#) is protected in tunnel mode by an outer IPsec header and, in this case, ESP, as shown in the following figure.

FIGURE 6-5 IPsec Packet Protected in Tunnel Mode



Encrypted

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

- For details on per-socket policy, see the [ipsec\(7P\)](#) man page.
- For an example of per-socket policy, see “[How to Use IPsec to Protect Web Server Communication With Other Servers](#)” on page 106.
- For more information about tunnels, see the [ipseccnf\(1M\)](#) man page.
- For an example of tunnel configuration, see “[How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode](#)” on page 112.

Virtual Private Networks and IPsec

The term [virtual private network \(VPN\)](#) is often used to describe a private, secure, point-to-point network that is built over a more public network, for example, the Internet. The point-

to-point network, or VPN, can be used to connect systems on private networks, or networks of systems on private networks together.

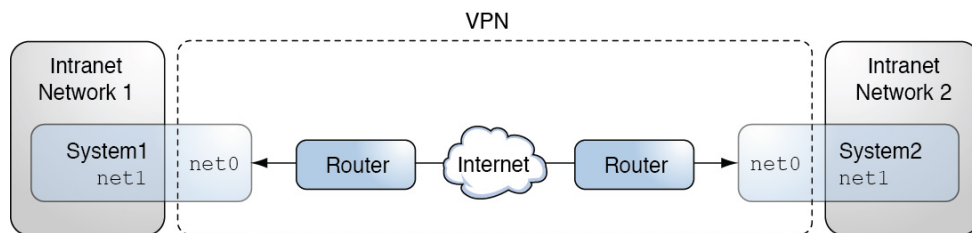
A configured [tunnel](#) is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see [“How to Create and Configure an IP Tunnel”](#) in [“Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2”](#).

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

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

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

FIGURE 6-6 Virtual Private Network



For a detailed example of the setup procedure, see [“How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode”](#) on page 112.

IPsec and FIPS 140

You can easily configure IPsec to comply with FIPS 140 requirements on a FIPS 140-enabled system. You are responsible for choosing only FIPS 140-validated algorithms to create keys and certificates. The procedures and examples in this guide use FIPS 140-approved algorithms except when the algorithm any is specified.

Note - If you have a strict requirement to use only FIPS 140-2 validated cryptography, you must be running the Oracle Solaris 11.1 SRU 5.5 release or the Oracle Solaris 11.1 SRU 3 release. Oracle completed a FIPS 140-2 validation against the Cryptographic Framework in these two specific releases. Oracle Solaris 11.2 builds on this validated foundation and includes software improvements that address performance, functionality, and reliability. Whenever possible, you should configure Oracle Solaris 11.2 in FIPS 140-2 mode to take advantage of these improvements.

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

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

For the definitive list of FIPS 140-validated algorithms for Oracle Solaris, see <http://www.oracle.com/technetwork/topics/security/140sp2061-2082028.pdf>. For a fuller discussion, see “Using a FIPS 140 Enabled System in Oracle Solaris 11.2”.

IPsec and NAT Traversal

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

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

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

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

The following RFCs describe NAT functionality and the limits of NAT-T. Copies of the RFCs are available at <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

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 [Stream Control Transmission Protocol \(SCTP\) RFC](#).

IPsec and Oracle Solaris Zones

IPsec is supported in zones. Each zone can have its own IPsec policy and IKE configuration. A zone can be treated like a separate host.

The exception is for shared-IP zones, which do not have their own IP stack. For shared-IP zones, the IPsec policy and IKE configuration are performed in the global zone. The IPsec policy rules for the shared-IP zone use the IP address that is assigned to that zone.

For more information, see [Chapter 1, “Oracle Solaris Zones Introduction,”](#) in “[Introduction to Oracle Solaris Zones](#)”.

IPsec and Virtual Machines

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

IPsec Configuration Commands and Files

[Table 6-2](#) describes the files, commands, and service identifiers that are used to configure and manage IPsec. For completeness, the table includes key management files, socket interfaces, and commands.

For more information about service identifiers, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in “[Managing System Services in Oracle Solaris 11.2](#)”.

For instructions on implementing IPsec on your network, see “[Protecting Network Traffic With IPsec](#)” on [page 101](#).

For more details about IPsec utilities and files, see [Chapter 12, “IPsec and Key Management Reference”](#).

TABLE 6-2 Selected IPsec Configuration Commands and Files

IPsec Command, File, or Service	Description	Man Page
<code>svc:/network/ipsec/ipsecalg</code> s	The SMF service that manages IPsec algorithms.	ipsecalgs(1M)
<code>svc:/network/ipsec/manual-key</code>	The SMF service that manages manually keyed IPsec SAs.	ipseckey(1M)
<code>svc:/network/ipsec/policy</code>	The SMF service that manages IPsec policy.	smf(5) , ipseconf(1M)
<code>svc:/network/ipsec/ike:ikev2</code> , <code>svc:/network/ipsec/ike:default</code>	The SMF service instances for the automatic management of IPsec SAs by using IKE.	smf(5) , in. ikev2d(1M) , in. iked(1M)
<code>/etc/inet/ipsecinit.conf</code> file	IPsec policy file. Used by the SMF <code>policy</code> service to configure IPsec policy at system boot.	ipseconf(1M)
<code>ipseconf</code> command	IPsec policy command. Useful for viewing and modifying the current IPsec policy, and for testing. Used by the SMF <code>policy</code> service to configure IPsec policy at system boot.	ipseconf(1M)
PF_KEY socket interface	Interface for the security associations database (SADB). Handles manual key management and automatic key management.	pf_key(7P)
<code>ipseckey</code> command	IPsec SAs keying command. <code>ipseckey</code> is a command-line front end to the PF_KEY interface. <code>ipseckey</code> can create, destroy, or modify SAs.	ipseckey(1M)
<code>/etc/inet/secret/ipseckeys</code> file	Contains manually keyed SAs. Used by the SMF <code>manual-key</code> service to configure SAs manually at system boot.	

IPsec Command, File, or Service	Description	Man Page
ipsecalgs command	<p>IPsec algorithms command. Useful for viewing and modifying the list of IPsec algorithms and their properties.</p> <p>Used by the SMF ipsecalgs service to synchronize known IPsec algorithms with the kernel at system boot.</p>	ipsecalgs(1M)
/etc/inet/ipsecalgs file	<p>Contains the configured IPsec mechanisms and algorithm definitions. This file is managed by the ipsecalgs command and must never be edited manually.</p>	
/etc/inet/ike/ikev2.config file	<p>IKEv2 configuration and policy file. Key management is based on rules and global parameters from this file. See “IKEv2 Utilities and Files” on page 217.</p>	ikev2.config(4)
/etc/inet/ike/config file	<p>IKEv1 configuration and policy file. By default, this file does not exist. Key management is based on rules and global parameters from this file. See “IKEv1 Utilities and Files” on page 220.</p> <p>If this file exists, the svc:/network/ipsec/ike:default service starts the IKEv1 daemon, in.iked.</p>	ike.config(4)

Configuring IPsec

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

- [“Protecting Network Traffic With IPsec” on page 101](#)
- [“Protecting a VPN With IPsec” on page 108](#)
- [“Additional IPsec Tasks” on page 116](#)

For overview information about IPsec, see [Chapter 6, “About IP Security Architecture”](#). For reference information about IPsec, see [Chapter 12, “IPsec and Key Management Reference”](#).

Note - These tasks assume that the systems are assigned static IP addresses and are running the network configuration profile `DefaultFixed`. If the `netadm list` command returns `Automatic`, see the [`netcfg\(1M\)`](#) man page for more information.

Protecting Network Traffic With IPsec

The procedures in this section enable you to secure traffic between two systems and to secure a web server. To protect a VPN, see [“Protecting a VPN With IPsec” on page 108](#). For additional procedures to manage IPsec and to use SMF commands with IPsec and IKE, see [“Additional IPsec Tasks” on page 116](#).

The following information applies to all IPsec configuration tasks:

- **IPsec and zones** – Each system is either a global zone or an exclusive-IP zone. For more information, see [“IPsec and Oracle Solaris Zones” on page 98](#).
- **IPsec and FIPS 140 mode** – As the IPsec administrator, you are responsible for choosing algorithms that are FIPS 140-validated for Oracle Solaris. The procedures and examples in this chapter use FIPS 140-approved algorithms except when the algorithm any is specified.
- **IPsec and RBAC** – To use roles to administer IPsec, see [Chapter 3, “Assigning Rights in Oracle Solaris,”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#). For an example, see [“How to Configure a Role for Network Security” on page 119](#).

- **IPsec and SCTP** – You can use IPsec to protect Streams Control Transmission Protocol (SCTP) associations, but caution must be used. For more information, see [“IPsec and SCTP” on page 98](#).
- **IPsec and Trusted Extensions labels** – On systems that are configured with the Trusted Extensions feature of Oracle Solaris, labels can be added to IPsec packets. For more information, see [“Administration of Labeled IPsec” in “Trusted Extensions Configuration and Administration”](#).
- **IPv4 and IPv6 addresses** – The IPsec examples in this guide use IPv4 addresses. Oracle Solaris supports IPv6 addresses as well. To configure IPsec for an IPv6 network, substitute IPv6 addresses in the examples. When protecting tunnels with IPsec, you can mix IPv4 and IPv6 addresses for the inner and outer addresses. This type of a configuration enables you to tunnel IPv6 over an IPv4 network, for example.

The following task map lists procedures that set up IPsec between one or more systems. The [ipseconf\(1M\)](#), [ipseckey\(1M\)](#), and [ipadm\(1M\)](#) man pages also describe useful procedures in their respective Examples sections.

TABLE 7-1 Protecting Network Traffic With IPsec Task Map

Task	Description	For Instructions
Secure traffic between two systems.	Protects packets from one system to another system.	“How to Secure Network Traffic Between Two Servers With IPsec” on page 102
Secure a web server by using IPsec policy.	Requires non-web traffic to use IPsec. Web clients are identified by particular ports that bypass IPsec checks.	“How to Use IPsec to Protect Web Server Communication With Other Servers” on page 106
Use IKE to automatically create keying material for IPsec SAs.	Recommended method of creating IPsec SAs.	“Configuring IKEv2” on page 135 and “Configuring IKEv1” on page 161
Set up a secure virtual private network (VPN).	Sets up IPsec between two systems across the Internet.	“Protecting a VPN With IPsec” on page 108
Set up manual key management.	Provides the raw data for IPsec SAs without using IKE.	“How to Manually Create IPsec Keys” on page 117

▼ How to Secure Network Traffic Between Two Servers With IPsec

This procedure assumes the following setup:

- The systems are assigned static IP addresses and are running the network configuration profile `DefaultFixed`. If the `netadm list` command returns `Automatic`, see the [netcfg\(1M\)](#) man page for more information.
- 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. This procedure uses IPv4 addresses.
- Each system is either a global zone or an exclusive-IP zone. For more information, see [“IPsec and Oracle Solaris Zones” on page 98](#).
- Each system encrypts traffic with the AES algorithm and authenticates it with SHA-2.

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

- Each system uses shared security associations.
With shared SAs, only one pair of SAs is needed to protect the two systems.

Note - To use IPsec with labels on a Trusted Extensions system, see the extension of this procedure in [“How to Apply IPsec Protections in a Multilevel Trusted Extensions Network” in “Trusted Extensions Configuration and Administration”](#).

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

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- In this administrative role, you can edit IPsec-related system files and create keys by using the `pfedit` command.
- To edit the `hosts` file, you must be in the root role or have explicit permission to edit that file. See [Example 7-7](#).

For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

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

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

a. On a system that is named `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.

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

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

For the syntax of IPsec policy entries and several examples, see the [ipsecconf\(1M\)](#) man page.

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

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

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

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

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

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

Follow one of the configuration procedures in “[Configuring IKEv2](#)” on page 135. For the syntax of the IKE configuration file, see the [ikev2.config\(4\)](#) man page. If you are communicating with a system that only supports the IKEv1 protocol, refer to “[Configuring IKEv1](#)” on page 161 and 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 117.

5. Verify the syntax of the IPsec policy file.

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

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

6. Refresh the IPsec policy.

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

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


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

7. Activate the keys for IPsec.

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

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

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

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

```
# svcadm restart ike:ikev2
```

If you manually configured keys in [Step 4](#), complete the procedure “[How to Manually Create IPsec Keys](#)” on [page 117](#) 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 123](#).

Example 7-1 Configuring IPsec Policy Remotely by Using an ssh Connection

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

1. The administrator configures the first system by performing [Step 1](#) through [Step 5](#) of “[How to Secure Network Traffic Between Two Servers With IPsec](#)” on [page 102](#).
2. In a different terminal window, the administrator uses the identically defined user name and ID to log in remotely with the `ssh` command.

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

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

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

5. The administrator enables IPsec policy on the first system by completing [Step 6](#) and [Step 7](#).

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

Example 7-2 Configuring IPsec Policy to Run in FIPS 140 Mode

In this example, the administrator configures the IPsec policy on a FIPS 140-enabled system to follow a site security policy that requires symmetric algorithms whose key length is at least 192 bits.

The administrator specifies two possible IPsec policies. The first policy specifies AES in CCM mode for encryption and authentication and the second policy specifies AES with 192-bit and 256-bit key lengths for encryption and SHA384 for authentication.

```
{laddr machine1 raddr machine2} ipsec {encr_algs aes-ccm(192...) sa shared} or ipsec  
{laddr machine1 raddr machine2} ipsec {encr_algs aes(192...) encr_auth_algs sha2(384) sa shared}
```

▼ How to Use IPsec to Protect Web Server Communication With Other Servers

On a system that runs a web server, you can use IPsec to protect all traffic except web client requests. The protected network traffic is typically between the web server and other backend servers.

In addition to allowing web clients to bypass IPsec, the IPsec policy in this procedure allows the server to make DNS client requests. All other traffic is protected by IPsec.

Before You Begin This procedure assumes that the steps in [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#) that configure IPsec on your two servers have been completed so that the following conditions are in effect:

- Each system is either a global zone or an exclusive-IP zone with a fixed address. For more information, see [“IPsec and Oracle Solaris Zones” on page 98](#).
- Communication with the web server is already protected by IPsec.
- Keying material is being generated by IKE.
- You have verified that packets are being protected.

A user with specific rights can run these commands without being root.

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- To edit IPsec-related system files and create keys, you use the `pfedit` command.

- To edit the `hosts` file, you must be in the root role or have explicit permission to edit that file.

For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell”](#) in [“Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

- 1. Determine which services need to bypass IPsec policy checks.**

For a web server, these services include TCP ports 80 (HTTP) and 443 (Secure HTTP). If the web server provides DNS name lookups, the server might also need to include port 53 for both TCP and UDP.

- 2. Add the web server policy to the IPsec policy file.**

Add the following lines to the `ipsecinit.conf` file:

```
# pfedit /etc/inet/ipsecinit.conf
...
# Web traffic that web server should bypass.
{port 80 ulp tcp dir both} bypass {}
{port 443 ulp tcp dir both} bypass {}

# Outbound DNS lookups should also be bypassed.
{rport 53 dir both} bypass {}

# Require all other traffic to use ESP with AES and SHA-2.
# Use a unique SA for outbound traffic from the port
{} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

This configuration allows only secure traffic to access the system, with the bypass exceptions that are described in [Step 1](#).

- 3. Verify the syntax of the IPsec policy file.**

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

- 4. Refresh the IPsec policy.**

```
# svcadm refresh ipsec/policy
```

- 5. Refresh the keys for IPsec.**

Restart the `ike` service.

```
# svcadm restart ike:ikev2
```

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

If you manually configured the keys, follow the instructions in [“How to Manually Create IPsec Keys” on page 117](#).

Your setup is complete.

6. (Optional) Enable a remote system to communicate with the web server for nonweb traffic.

Add the following lines to a remote system's `/etc/inet/ipsecinit.conf` file:

```
## Communicate with web server about nonweb stuff
##
{raddr webserver} ipsec {encr_algs aes encr_auth_algs sha512 sa shared}
```

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

```
remote-system # ipsecconf -c /etc/inet/ipsecinit.conf
remote-system # svcadm refresh ipsec/policy
```

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

7. (Optional) Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.

```
# ipsecconf -L -n
```

Protecting a VPN With IPsec

You can use IPsec to protect a VPN. For background, see [“Transport and Tunnel Modes in IPsec” on page 93](#). The examples and procedures in this section use IPv4 addresses, but the examples and procedures apply to IPv6 VPNs as well. For a short discussion, see [“Protecting Network Traffic With IPsec” on page 101](#).

For examples of IPsec policies for tunnel mode, see [“Examples of Protecting a VPN With IPsec by Using Tunnel Mode” on page 109](#).

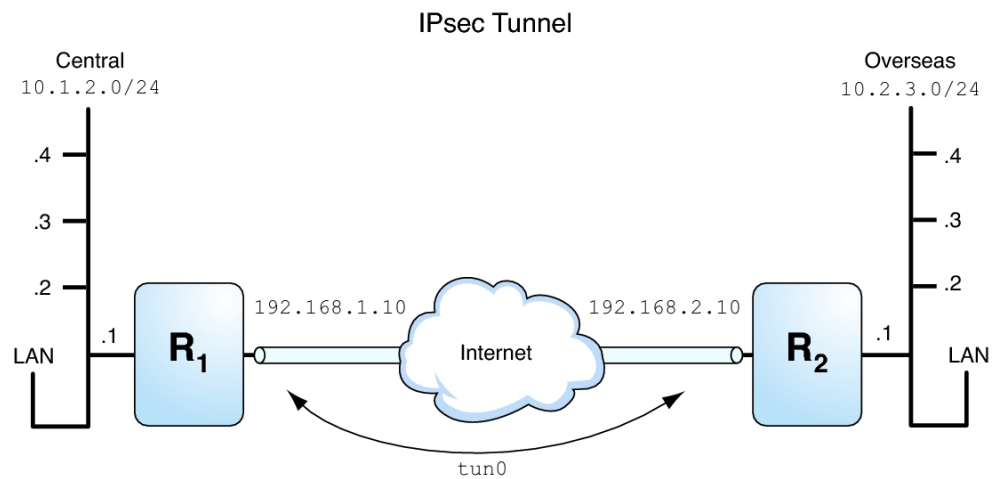
Examples of Protecting a VPN With IPsec by Using Tunnel Mode

The tunnel in the following illustration is configured for all subnets of the LANs as follows:

```
## Tunnel configuration for ##
# Tunnel name is tun0
# Intranet point for the source is 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
```

FIGURE 7-1 Tunnel Protected by IPsec



The following examples are based on the illustration.

EXAMPLE 7-3 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 7-4 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.

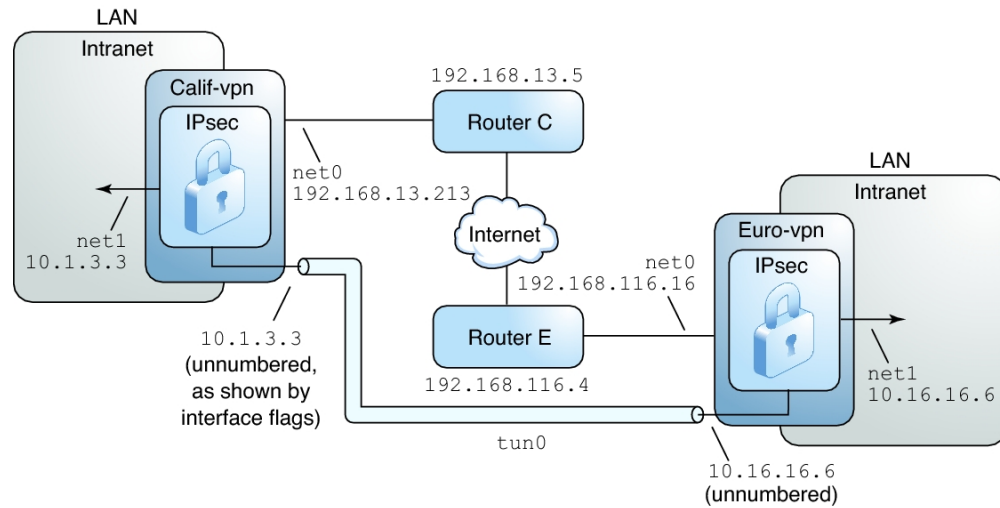
```
## IPsec policy ##  
{tunnel tun0 negotiate tunnel laddr 10.1.2.0/24 raddr 10.2.3.0/24}  
ipsec {encr_algs aes encr_auth_algs sha512 shared}
```

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

The procedures in this section assume the following setup. For a depiction of the network, see [Figure 7-2](#).

- Each system is using an IPv4 address space.
These procedures also work with IPv6 addresses or a combination of IPv4 and IPv6 addresses.
- Each system has two interfaces. The net0 interface connects to the Internet. In this example, Internet IP addresses begin with 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 encryption with the AES algorithm. The AES algorithm uses a 128-bit or 256-bit key.
- Each system requires ESP authentication with the SHA-2 algorithm. In this example, the SHA-2 algorithm uses a 512-bit key.
- Each system can connect to a router that has direct access to the Internet.
- Each system uses shared security associations.

The following illustration shows the configuration parameters used in the procedures.

FIGURE 7-2 Sample VPN Between Offices Connected Across the Internet

The configuration parameters are listed in the following table.

Parameter	Europe	California
System name	euro-vpn	calif-vpn
System intranet interface	net1	net1
System intranet address, the default route to the other network	10.16.16.6	10.1.3.3
System intranet address object	net1/inside	net1/inside
System Internet interface	net0	net0
System Internet address	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 [“Administering IP Tunnels”](#) in [“Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.2”](#). For information about address

objects, see [“How to Configure an IPv4 Interface”](#) in [“Configuring and Administering Network Components in Oracle Solaris 11.2 ”](#) and the `ipadm(1M)` man page.

▼ How to Protect the Connection Between Two LANs With IPsec in Tunnel Mode

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

This procedure extends the procedure [“How to Secure Network Traffic Between Two Servers With IPsec”](#) on page 102. The setup is described in [“Description of the Network Topology for the IPsec Tasks to Protect a VPN”](#) on page 110.

For a fuller description of the reasons for running particular commands, see the corresponding steps in [“How to Secure Network Traffic Between Two Servers With IPsec”](#) on page 102.

Note - Perform the steps in this procedure on both systems.

In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

Note - To use IPsec in tunnel mode with labels on a Trusted Extensions system, see the extension of this procedure in [“How to Configure a Tunnel Across an Untrusted Network”](#) in [“Trusted Extensions Configuration and Administration ”](#).

Before You Begin Each system is either a global zone or an exclusive-IP zone. For more information, see [“IPsec and Oracle Solaris Zones”](#) on page 98.

A user with specific rights can run these commands without being root.

- To run configuration commands, you must become an administrator who is assigned the Network IPsec Management rights profile.
- To edit IPsec-related system files and create keys, you use the `pfedit` command.
- To edit the `hosts` file, you must be in the `root` role or have explicit permission to edit that file.

For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2 ”](#).

If you administer remotely, see [Example 7-1](#) and “[How to Remotely Administer ZFS With Secure Shell](#)” in “[Managing Secure Shell Access in Oracle Solaris 11.2](#)” for secure remote login instructions.

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
```

Disabling 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. Enable IP strict multihoming.

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

Enabling IP strict multihoming requires that packets for one of the system's destination addresses arrive at the correct destination address.

When the `hostmodel` parameter is set to `strong`, packets that arrive on a particular interface must be addressed to one of the local IP addresses of that interface. All other packets, even packets that are addressed to other local addresses of the system, are dropped.

c. Verify that most network services are disabled.

Verify that the `ssh` service is running.

```
% svcs | grep network
...
online          Aug_09   svc:/network/ssh:default
```

2. Add the IPsec policy for the VPN to the `/etc/inet/ipsecinit.conf` file.

For additional examples, see “[Examples of Protecting a VPN With IPsec by Using Tunnel Mode](#)” on [page 109](#).

In this policy, IPsec protection is not required between systems on the local LAN and the internal IP address of the gateway, so a `bypass` statement is added.

a. On the `euro-vpn` system, add the following entry to the `ipsecinit.conf` file:

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 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, add the following entry to 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 IKEv2” on page 135](#). For the syntax of the IKE configuration file, see the [`ikev2.config\(4\)` man page](#). If you are communicating with a system that only supports the IKEv1 protocol, refer to [“Configuring IKEv1” on page 161](#) and 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 117](#).

4. Verify the syntax of the IPsec policy file.

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

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

5. Refresh the IPsec policy.

```
# svcadm refresh ipsec/policy
```

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

```
# svcadm enable ipsec/policy
```

6. Create and configure the tunnel, `tun0`.

The following commands configure the internal and external interfaces, create the `tun0` tunnel, and assign IP addresses to the tunnel.

a. On the `calif-vpn` system, create the tunnel and configure it.

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.1.3.3 net1/inside
# dladm create-iptun -T ipv4 -a local=192.168.13.213,remote=192.168.116.16 tun0
```

```
# ipadm create-ip tun0
# ipadm create-addr -T static \
-a local=10.1.3.3,remote=10.16.16.6 tun0/v4tunaddr
```

The first command creates the IP interface `net1`. The second command adds addresses to `net1`. The third command creates the IP interface `tun0`. The fourth command adds IP addresses that are encapsulated in the tunnel link. For more information, see the [dladm\(1M\)](#) and [ipadm\(1M\)](#) man pages.

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

```
# ipadm create-ip net1
# ipadm create-addr -T static -a local=10.16.16.6 net1/inside
# dladm create-iptun -T ipv4 -a local=192.168.116.16,remote=192.168.13.213 tun0
# ipadm create-ip tun0
# ipadm create-addr -T static \
-a local=10.16.16.6,remote=10.1.3.3 tun0/v4tunaddr
```

Note - The `-T` option to the `ipadm` command specifies the type of address to create. The `-T` option to the `dladm` command specifies the tunnel.

For information about these commands, see the [dladm\(1M\)](#) and [ipadm\(1M\)](#) man pages, and “How to Configure an IPv4 Interface” in “Configuring and Administering Network Components in Oracle Solaris 11.2”. For information about customized names, see “Network Devices and Datalink Naming in Oracle Solaris” in “Configuring and Administering Network Components in Oracle Solaris 11.2”.

7. On each system, configure forwarding.

```
# ipadm set-ifprop -m ipv4 -p forwarding=on net1
# ipadm set-ifprop -m ipv4 -p forwarding=on tun0
# ipadm set-ifprop -m ipv4 -p forwarding=off net0
```

IP forwarding means that packets that arrive from somewhere else can be forwarded. IP forwarding also means that packets that leave this interface might have originated somewhere else. To successfully forward a packet, both the receiving interface and the transmitting interface must have IP forwarding enabled.

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

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

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

Even if `net0` has IP forwarding disabled, a routing protocol implementation might still advertise the interface. For example, the `in.routed` protocol might still advertise that `net0` is available to forward packets to its peers inside the intranet. By setting the interface's private flag, these advertisements are prevented.

9. Restart the network services.

```
# svcadm restart svc:/network/initial:default
```

10. Manually add a default route over the `net0` interface.

The default route must be a router with direct access to the Internet.

a. On the `calif-vpn` system, add the following route:

```
# route -p add net default 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.

Additional IPsec Tasks

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

TABLE 7-2 Additional IPsec Tasks Task Map

Task	Description	For Instructions
Create or replace IPsec SAs manually.	Provides the raw data for IPsec SAs:	“How to Manually Create IPsec Keys” on page 117
Create a Network Security role.	Creates a role that can set up a secure network, but has fewer powers than the <code>root</code> role.	“How to Configure a Role for Network Security” on page 119
Create a rights profile that can handle all network management tasks.	Creates a role that can perform network management but has fewer powers than the <code>root</code> role.	Example 7-7

Task	Description	For Instructions
Check that IPsec is protecting the packets.	Examines snoop output for specific headers that indicate how the IP packets are protected.	“How to Verify That Packets Are Protected With IPsec” on page 123
Manage IPsec and keying material as a set of SMF services.	Enables, disables, refreshes, and restarts services. Also changes the property values of services.	“Viewing IPsec and Manual Key Service Properties” on page 201

▼ How to Manually Create IPsec Keys

The following procedure provides the IPsec keys for when you are not using only IKE for key management.

IPsec SAs that are added by using the `ipseckey` command are not persistent over system reboot. For persistent IPsec SAs, add entries to the `/etc/inet/secret/ipseckey` file.



Caution - If you must use manual keying, take great care to ensure that the keys that you generate are secure. These are the actual keys used to secure the data.

Before You Begin You must be in the global zone to manually manage keying material in a shared-IP zone. For an exclusive-IP zone, you configure the keying material in that exclusive-IP zone.

You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

1. Generate the keys for the IPsec SAs.

The keys must support a specific policy in the `ipseccinit.conf` file. For example, you might use the policy from [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#):

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

This policy uses the AES and SHA-2 algorithms.

a. Determine the keys that you require.

You need to generate keys for `aes`, `sha512`, and the [security parameter index \(SPI\)](#) for the SA:

- Two hexadecimal random numbers as the value for the SPI. One number is for outbound traffic. One number is for inbound traffic. Each number can be up to eight characters long.

- Two hexadecimal random numbers for the SHA-2 authentication algorithm. Each number must be 512 characters long. One number is for `dst enigma`. One number is for `dst partym`.
- Two hexadecimal random numbers for the AES encryption algorithm. Each number must be 128 characters long. One number is for `dst enigma`. One number is for `dst partym`.

Note - The `ipsecalgs -l` command displays the key sizes of the algorithms. Follow this procedure when using manual keys, that is, use the SHA512 and AES algorithms. Do not use weak algorithms, the combined mode algorithms, or the GMAC algorithms for manual keys.

b. Generate the required keys.

- If you have a random number generator at your site, use the generator.
- Use the `pktool` command, as shown in [“How to Generate a Symmetric Key by Using the pktool Command”](#) in [“Managing Encryption and Certificates in Oracle Solaris 11.2”](#) and the IPsec example in that section.

2. Add the keys to the manual keys file for IPsec.

a. Edit the `/etc/inet/secret/ipseckeys` file on the 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 aes \
    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 aes \
    auth_alg sha512 \
    encrkey dd325c5c137fb4739a55c9b3a1747baa... \
    authkey ad9ced7ad5f255c9a8605fba5eb4d2fd...
```

b. Protect the file with read-only permissions.

```
# chmod 400 /etc/inet/secret/ipseckeys
```

If you used the `pfedit -s` command to create the `ipseckeys` file, then the permissions are correctly set. For more information, see the [pfedit\(1M\)](#) man page.

c. Verify the syntax of the file.

```
# ipseckey -c /etc/inet/secret/ipseckeys
```

Note - The keys on the two systems *must* be identical.

3. Activate the keys for IPsec.

■ **If the `manual-key` service is not enabled, enable it.**

```
% svcs manual-key
STATE          STIME    FMRI
disabled      Apr_10   svc:/network/ipsec/manual-key:default
# svcadm enable ipsec/manual-key
```

■ **If the `manual-key` service is enabled, refresh it.**

```
# svcadm refresh ipsec/manual-key
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Configure a Role for Network Security

If you are using the rights feature of Oracle Solaris to administer your systems, use this procedure to provide a network management role or network security role.

Before You Begin You must assume the `root` role to create and assign a role. Regular users can list and view the contents of available rights profiles.

1. List the available network-related rights profiles.

```
% getent prof_attr | grep Network | more
...
Network Management:RO::Manage the host and network configuration...
Network Security:RO::Manage network and host security...:profiles=Network Wifi
Security,Network Link Security,Network IPsec Management...
```

```
Network Wifi Management:RO::Manage wifi network configuration...
Network Wifi Security:RO::Manage wifi network security...
Network Link Security:RO::Manage network link security...
Network IPsec Management:RO::Manage IPsec and IKE...
System Administrator:RO::Can perform most non-security administrative tasks:
profiles=...Network Management...
Information Security:RO::Maintains MAC and DAC security policies:
profiles=...Network Security...
```

The Network Management profile is a supplementary profile in the System Administrator profile. If you have included the System Administrator rights profile in a role, then that role can execute the commands in the Network Management profile.

2. List the commands in the Network Management rights profile.

```
% profiles -p "Network Management" info
...
cmd=/usr/sbin/dladm
cmd=/usr/sbin/dlstat
...
cmd=/usr/sbin/svcadm
cmd=/usr/sbin/svccfg
cmd=/usr/sbin/dumpcap
```

3. Decide the scope of the network security roles at your site.

Use the definitions of the rights profiles in [Step 1](#) to guide your decision.

- To create a role that handles all network security, use the Network Security rights profile.
- To create a role that handles IPsec and IKE only, use the Network IPsec Management rights profile.
- To create a role that handles network management and security, use the Network Security or the Network IPsec Management rights profile, in addition to the Network Management profile.

4. Create the role and assign the role to one or more users.

For the steps, see [“Creating a Role”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#) and [Example 7-7](#).

Example 7-5 Creating and Assigning a Network Management and Security Role

In this example, the administrator assigns to a role two rights profiles, Network Management and Network Security. Then the administrator assigns the role to a trusted user.

```
# roleadd -c "Network Mgt and Security" \
-S ldap -K profiles="Network Management Plus" netmgtsec
# passwd netmgtsec
New Password: xxxxxxxx
Confirm password: xxxxxxxx
# usermod -R netmgtsec jdoe
```


The rights in the profiles are available to the user `jdoe` after `jdoe` assumes the `netmgtsec` role.

```
% su - netsecmgt
Password: xxxxxxxx
#
```

Example 7-6 Dividing Network Security Responsibilities Between Roles

In this example, the administrator divides network security responsibilities between two roles. One role administers Wifi and link security and another role administers IPsec and IKE. Each role is assigned to three people, one person per shift.

The roles are created by the administrator as follows:

1. The administrator names the first role LinkWifi.
2. The administrator assigns the Network Wifi, Network Link Security, and Network Management rights profiles to the role.
3. The administrator assigns the LinkWifi role to the appropriate users.
4. The administrator names the second role IPsec Administrator.
5. The administrator assigns the Network IPsec Management and the Network Management rights profiles to the role.
6. The administrator assigns the IPsec Administrator role to the appropriate users.

Example 7-7 Enabling a Trusted User to Configure and Manage IPsec

In this example, the administrator gives one user responsibility for configuring and managing IPsec.

In addition to the Network Management and IPsec Network Management rights profiles, the administrator gives the user the ability to edit the `hosts` file and the ability to read the logs.

1. The administrator creates two rights profiles, one for editing files and the other for reading logs.

```
# profiles -p -S LDAP "Hosts Configuration"
profiles:Network Configuration> set desc="Edits root-owned network files"
...Configuration> add auth=solaris.admin.edit/etc/hosts
...Configuration> commit
...Configuration> end
...Configuration> exit

# profiles -p -S LDAP "Read Network Logs"
profiles:Read Network Logs> set desc="Reads root-owned network log files"
...Logs> add cmd=/usr/bin/more
...Logs:more>set privs={file_dac_read}:/var/user/ikeuser/*
```

```

...Logs:more>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:more> set privs={file_dac_read}:/etc/inet/ike/*
...Logs:more> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:more>end
...Logs> add cmd=/usr/bin/tail
...Logs:tail>set privs={file_dac_read}:/var/user/ikeuser/*
...Logs:tail>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:tail>set privs={file_dac_read}:/etc/inet/ike/*
...Logs:tail> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:tail>end
...Logs> add cmd=/usr/bin/page
...Logs:page>set privs={file_dac_read}:/var/user/ikeuser/*
...Logs:page>set privs={file_dac_read}:/var/log/ikev2/*
...Logs:page>set privs={file_dac_read}:/etc/inet/ike/*
...Logs:page> set privs={file_dac_read}:/etc/inet/secret/*
...Logs:page>end
...Logs> exit

```

The rights profile enables the user to use the more, tail, and page commands to read the logs. The cat and head commands cannot be used.

2. The administrator creates the rights profile that enables the user to perform all configuration and management tasks for IPsec and its keying services.

```

# profiles -p "Site Network Management"
profiles:Site Network Management> set desc="Handles all network files and logs"
...Management> add profiles="Network Management"
...Management> add profiles="Network IPsec Management"
...Management> add profiles="Hosts Configuraton"
...Management> add profiles="Read Network Logs"
...Management> commit; end; exit

```

3. The administrator creates a role for the profile, assigns it a password, and assigns the role to a trusted user who understands networking and security.

```

# roleadd -S LDAP -c "Network Management Guru" \
-m -K profiles="Site Network Management" netadm
# passwd netadm
Password: xxxxxxxx
Confirm password: xxxxxxxx
# usermod -S LDAP -R +netadm jdoe

```

4. Out of band, the administrator supplies jdoe with the role password.

▼ How to Verify That Packets Are Protected With IPsec

To verify that packets are protected, test the connection with the snoop command. The following prefixes can appear in the snoop output:

- AH: Prefix indicates that AH is protecting the headers. You see this prefix if you used `auth_alg` to protect the traffic.
- ESP: Prefix indicates that encrypted data is being sent. You see this prefix if you used `encr_auth_alg` or `encr_alg` to protect the traffic.

Before You Begin You must have access to both systems to test the connection.

You must assume the root role to create the snoop output. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

1. On one system, such as `partym`, assume the root role.

```
% su -
Password: xxxxxxxx
#
```

2. (Optional) Display the details of the SAs.

```
# ipseckey dump
```

This output indicates which SPI values match the SAs that are used, which algorithms were used, the keys, and so on.

3. On this system, prepare to snoop packets from a remote system.

In a terminal window on `partym`, snoop the packets from the `enigma` system.

```
# snoop -d net0 -o /tmp/snoop_capture enigma
Using device /dev/e1000g (promiscuous mode)
```

4. 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.

```
partym% ssh enigma
Password: xxxxxxxx
enigma% su -
Password: xxxxxxxx
enigma# ping partym
```

5. Examine the snoop output.

```
partym# snoop -i /tmp.snoop_capture -v
```

You can also load the snoop output into the Wireshark application. For more information, see [“How to Prepare IPsec and IKE Systems for Troubleshooting” on page 194](#) and [“snoop Command and IPsec” on page 215](#).

In the file, you should see output that includes AH and ESP information after the initial IP header information. AH and ESP information that resembles the following shows that packets are being protected:

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 4e0e
IP:   Source address = 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 -----
...
```

About Internet Key Exchange

Internet Key Exchange (IKE) automates key management for IPsec. This chapter contains the following information about IKE:

- [“Introduction to IKE” on page 125](#)
- [“IKEv2 Protocol” on page 131](#)
- [“IKEv1 Protocol” on page 132](#)

For instructions on implementing the latest version of the IKE protocol, see [Chapter 9, “Configuring IKEv2”](#). To continue to use IKEv1, see [Chapter 10, “Configuring IKEv1”](#). For reference information, see [Chapter 12, “IPsec and Key Management Reference”](#). For information about IPsec, see [Chapter 6, “About IP Security Architecture”](#).

Introduction to IKE

The management of keying material for IPsec security associations (SAs) is called *key management*. Automatic key management requires a secure channel of communication for the creation, authentication, and exchange of keys. Oracle Solaris uses Internet Key Exchange (IKE) to automate key management. IKE eliminates administrative overhead and the security risk of manually distributing secret keys.

IKE can take advantage of available hardware cryptographic acceleration and key storage. Hardware cryptographic accelerators permit CPU-intensive key operations to be handled off the system. Key storage on hardware provides an additional layer of protection.

Oracle Solaris supports two versions of the IKE protocol.

- IKE Version 2 (IKEv2), which is based on [Internet Key Exchange Protocol Version 2 \(IKEv2\), RFC 5996](#)
- IKE Version 1 (IKEv1), which is based on [The Internet Key Exchange \(IKE\), RFC 2409](#)

IKE Concepts and Terminology

The following concepts and terms are common to both versions of IKE. They might be implemented differently in the two versions.

- **Key negotiation and exchange** – The exchange of keying material and the authentication of the peer's identity in a secure manner. The process uses asymmetric cryptographic algorithms. The two main methods are the RSA and the Diffie-Hellman protocols.
IKE creates and manages the IPsec SAs between systems that are running an IKE daemon. IKE negotiates a secure channel that protects the transmission of keying material. The daemon creates the keys from a random number generator by using the `/dev/random` device. The daemon changes the keys at a configurable rate. The keying material is available to algorithms that are specified in the configuration file for IPsec policy, `ipsecinit.conf`.
- **Diffie-Hellman (DH) algorithm** – A key exchange algorithm that allows two systems to securely generate a shared secret over an insecure channel.
- **RSA algorithm** – An asymmetric key algorithm that is used to authenticate the identity of peer systems, typically by proving ownership of an X.509 certificate. The algorithm is named for its three creators: Rivest, Shamir, and Adleman.
Alternatively, [DSA](#) or [ECDSA](#) algorithms may be used for this purpose.
- **Perfect forward secrecy (PFS)** – In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys. Therefore, PFS can prevent the decryption of previously recorded traffic.
- **Oakley group** – Used to negotiate PFS. See Section 6 of the [The Internet Key Exchange \(IKE\) RFC](#).
- **IKE policy** – The set of IKE rules which define the acceptable parameters that an IKE daemon uses when attempting to set up a secure key exchange channel with a peer system. This is called an IKE SA in IKEv2 or Phase 1 in IKEv1.
The parameters include algorithms, key sizes, Oakley groups, and authentication method. The Oracle Solaris IKE daemons support preshared keys and certificates as authentication methods.

How IKE Works

A system that is running an IKE daemon can negotiate the parameters needed to create a security association (SA) between this system and another system that is running the IKE daemon. The protocol that is used to negotiate this SA and subsequent IPsec SAs is known as IKE. This version of Oracle Solaris supports version 1 (IKEv1) and version 2 (IKEv2) of the IKE protocol.

The IKE security association (also known as the ISAKMP or Phase 1 SA in IKEv1) secures further protocol exchanges between these two IKE systems. These exchanges negotiate cryptographic algorithms, IPsec policy, and other parameters needed to create IPsec SAs.

Systems that are running an IKE daemon can also be configured to negotiate IPsec SAs on behalf of other systems. When configured in this manner, the systems are referred to as *security gateways*. If the IKE negotiation is successful, the IPsec SAs can be used to protect network packets.

Note - In Oracle Solaris 11.2, IKEv2 uses cryptographic algorithms from the Cryptographic Framework that are validated for FIPS 140-2, Level 1, but IKEv1 does not. By default, FIPS 140 is not enabled. For a comparison of the features of the two versions, see [“Comparison of IKEv2 and IKEv1” on page 130](#). To enable FIPS 140-2 mode, see [“How to Create a Boot Environment with FIPS 140 Enabled” in “Managing Encryption and Certificates in Oracle Solaris 11.2”](#).

If you have a strict requirement to use only FIPS 140-2 validated cryptography, you must be running the Oracle Solaris 11.1 SRU 5.5 release or the Oracle Solaris 11.1 SRU 3 release. Oracle completed a FIPS 140-2 validation against the Cryptographic Framework in these two specific releases. Oracle Solaris 11.2 builds on this validated foundation and includes software improvements that address performance, functionality, and reliability. Whenever possible, you should configure Oracle Solaris 11.2 in FIPS 140-2 mode to take advantage of these improvements.

The parameters that are negotiated to create the IKE SA include the cryptographic algorithms that protect the IKE exchanges and some authentication material. The authentication material is used to determine whether the packets that contain the IKE protocol exchanges can be trusted. Trust means that the packets come from a trusted system and not from a system that is pretending to be that system.

Oracle Solaris supports two types of authentication material for IKE, preshared keys and public key certificates.

IKE With Preshared Key Authentication

A preshared key is string of hex or ASCII characters that only the two IKE systems know. The keys are called *preshared* because both endpoints must know the value of the key before the IKE exchange. This key must be part of the IKE configuration on both systems. The preshared key is used in the generation of the IKE payloads, which make up the packets that implement the IKE protocol. The system that processes these IKE payloads uses the same key to authenticate the payloads that it receives.

The preshared key is not exchanged between the IKE endpoints by using the IKE protocol. Typically, the key is shared with the peer system over a different medium, such as a phone call.

The preshared key on the peers that use this authentication method must be identical. The keys are stored in a file on each system.

IKE With Public Key Certificates

Public key certificates and their trust chains provide a mechanism to digitally identify a system without having to manually exchange any secret information. Therefore, public key certificates are more secure than preshared keys.

A public key certificate is a blob of data that encodes a public key value, some information about the generation of the certificate, such as a name and who signed it, a hash or checksum of the certificate, and a digital signature of the hash. Together, these values form the certificate. The digital signature ensures that the certificate has not been modified.

A public key is a value that is mathematically derived from another value, called the *private key*. The mathematical algorithm that derives the public key from the private key makes retrieving the private key from the public key impractical. Therefore, public key certificates can be freely shared. Examples of algorithms that are used to derive public keys include [RSA](#) and Elliptic Curve.

A digital signature is the result of passing the certificate contents through a digital signing algorithm such as [RSA](#), [DSA](#) or [ECDSA](#). These algorithms use a private signing key, which is not part of the certificate, and produce a digital signature. The signature is appended to the certificate. Again, calculating the signing key from the certificate contents and the signature is impractical. More to the point, the certificate signature and hence the certificate contents can be easily verified by using a public value, which was derived from the signing key.

A certificate can be self-signed, in which case the signature of the certificate can be verified by the certificate's public key, or it can be signed by a different entity. When a different entity signs the certificate, the public key value that is used to verify the certificate is also distributed as a public key certificate. This second certificate will be signed by a [certificate authority \(CA\)](#) which is trusted, or by an intermediary. The intermediary is ultimately trusted by the signing entity, that is, the root CA or [trust anchor](#).

These public key certificate components, plus the procedures and structures that implement them are often referred to as a public key infrastructure ([PKI](#)). The scope of an organization's PKI can vary. A simple PKI could consist of a CA that signs a few certificates for local use. A more extensive PKI would use a globally recognized trust anchor as the authoritative CA.

Using Public Key Certificates in IKE

This section describes the overall steps to create and use public key certificates in IKE. For the specific procedures, see [“Configuring IKEv2 With Preshared Keys” on page 136](#) and [“Configuring IKEv1 With Preshared Keys” on page 162](#).

1. To use either a self-signed certificate or a certificate from a certificate authority (CA), you first generate a public/private key pair.
 - For a self-signed certificate, IKE peers then exchange these certificates, verify out of band that the certificates are genuine, and then import the peers' certificates into the local keystore. The keystore then contains the original self-signed certificate plus the imported certificates.
 - For certificates from a CA, you perform several more steps. When you generate the public/private key pair, you also generate a certificate signing request (CSR). A CSR contains the public key and *identifiers*. A typical identifier is a [distinguished name \(DN\)](#), for example:

```
DN="O=Example\, Inc, OU=qa, L=Silicon Valley, ST=CA, CN=enigma"
```

Tip - Create a DN or other identifier that is as specific as possible to reduce the possibility of matching another certificate's identifier.

2. Send the CSR to the CA for signature.

In a typical process, you paste the CSR into a web form and submit the form to the CA. The CA might send more than one signed certificate to you.
3. Get the signed certificates from the CA, then import them into your IKEv2 keystore or IKEv1 database.

You must import all the certificates that the CA sends. These certificates comprise a “chain of trust” from the trust anchor, or root CA, to your individually identified signed certificate.
4. Repeat the process on an IKE peer.
5. Use the certificates in IKE rules.

You specify the certificate by an identifier, such as a DN. For CA-signed certificates, you can configure IKE to accept any certificate that is signed by a particular CA.

Handling Revoked Certificates

A signed certificate is trusted as valid because the signing authority assures its validity. If a certificate is compromised or otherwise determined as invalid, the CA will revoke it.

CAs maintain a list of revoked certificates, often called the [certificate revocation list \(CRL\)](#). You can use the Online Certificate Status Protocol (OCSP) to dynamically check the status of a certificate. Some public key certificates have URIs embedded in them. They identify a web location where you can check the CRL or the web location of an OCSP server.

For more information, see [RFC 2459: Certificate and CRL Profile](#) and [RFC 2560: Online Certificate Status Protocol - OCSP](#).

Coordinating Time on Systems That Use Public Certificates

Public key certificates contain the date and time of issue and the time that they remain valid. Therefore, the clocks on systems that generate and use certificates must be accurate. The Network Time Protocol (NTP) software can be used to synchronize the clocks on systems. NTP public domain software from the University of Delaware is included in the Oracle Solaris release. Documentation is available from the [NTP Documentation](#) web site. You can also install the `service/network/ptp` package to configure the Precision Time Protocol (PTP) service. See [IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems](#).

Comparison of IKEv2 and IKEv1

The following table compares the implementation of the IKEv2 and IKEv1 versions on an Oracle Solaris system.

TABLE 8-1 IKEv2 and IKEv1 Implementation in Oracle Solaris

Feature	IKEv2	IKEv1
Certificate chain of trust	Implicit based on objects in keystore	<code>cert_trust</code> parameter in <code>ike/config</code> file
Certificate creation	<code>ikev2cert</code> command	<code>ikecert certlocal</code> command
Certificate import	<code>ikev2cert import</code> command can import certificates and keys into PKCS #11 keystore	<code>ikecert certdb</code> command can import standalone certificates into IKE keystore
Certificate owner	<code>ikeuser</code>	<code>root</code>
Certificate policy file	<code>kmf-policy.xml</code>	Some policy in <code>ike/config</code> file
Certificate storage	PKCS #11 softtoken library	Local IKEv1 databases
Configuration file directory	<code>/etc/inet/ike/</code>	<code>/etc/inet/ike/</code> and <code>/etc/inet/secret/</code>
Configuration owner	<code>ikeuser</code> account	<code>root</code> account
Daemon	<code>in.ikev2d</code>	<code>in.iked</code>
FIPS 140 algorithms for traffic between daemons [†]	IKE SAs use the Cryptographic Framework	Not all exchanges use the Cryptographic Framework
FIPS 140 algorithms for IPsec traffic [†]	Use the Cryptographic Framework	Use the Cryptographic Framework
IKE policy file	<code>ike/ikev2.config</code>	<code>ike/config</code>
IKE preshared keys	<code>ike/ikev2.preshared</code>	<code>secret/ike.preshared</code>

Feature	IKEv2	IKEv1
NAT port	UDP port 4500	UDP port 4500
Port	UDP port 500	UDP port 500
Rights profile	Network IPsec Management	Network IPsec Management
Service name (FMRI)	svc:/ipsec/ike:ikev2	svc:/ipsec/ike:default

[†]The Cryptographic Framework feature of Oracle Solaris 11.1 SRU 5.5 and SRU 3 is validated for FIPS 140-2, Level 1. If FIPS 140 mode is enabled and the Cryptographic Framework is being used, then FIPS 140-validated algorithms are used. By default, FIPS 140 mode is not enabled.

IKEv2 Protocol

This section covers the implementation of IKEv2. For IKEv1 information, see [“IKEv1 Protocol” on page 132](#). For a comparison, see [“Comparison of IKEv2 and IKEv1” on page 130](#). For information that applies to both protocols, see [“Introduction to IKE” on page 125](#). Oracle Solaris supports both versions of the IKE protocol simultaneously.

The IKEv2 daemon, `in.ikev2d`, negotiates and authenticates keying material for IPsec SAs. See the [`in.ikev2d\(1M\)`](#) man page.

IKEv2 Configuration Choices

The `/etc/inet/ike/ikev2.config` configuration file contains the configuration for the `in.ikev2d` daemon. The configuration consists of a number of rules. Each entry contains parameters such as algorithms and authentication data that this system can use with a similarly configured IKEv2 peer.

The `in.ikev2d` daemon supports preshared keys (PSK) and public key certificates for identity.

The [`ikev2.config\(4\)`](#) man page provides sample rules. Each rule must have a unique label. The following is a list of the descriptive labels of sample rules from the man page:

- IP identities and PSK auth
- IP address prefixes and PSK auth
- IPv6 address prefixes and PSK auth
- Certificate auth with DN identities
- Certificate auth with many peer ID types
- Certificate auth with wildcard peer IDs
- Override transforms
- Mixed auth types

- Wildcard with required signer

Note - A preshared key can be used with any one of many peer ID types, including IP addresses, DNs, FQDNs, and email addresses.

IKEv2 Policy for Public Certificates

The `kmf-policy.xml` file contains the certificate validation policy for IKEv2. The `kmfcfg dbfile=/etc/inet/ike/kmf-policy.xml policy=default` command is used to modify certificate validation policy. Typical modifications include the use of OCSP and CRLs, and the duration of network timeouts during certificate verification. Additionally, the policy enables an administrator to modify various aspects of certificate validation, such as validity date enforcement and key usage requirements. Loosening the default requirements for certificate validation is not recommended.

IKEv1 Protocol

The following sections provide an overview of IKEv1. IKEv1 is superseded by IKEv2, which offers faster, secured key management. For information about IKEv2, see [“IKEv2 Protocol” on page 131](#). For a comparison, see [“Comparison of IKEv2 and IKEv1” on page 130](#). For information that is common to both protocols, see [“Introduction to IKE” on page 125](#). IKEv1 and IKEv2 can run simultaneously and negotiate with their peer protocol on other systems.

IKEv1 Key Negotiation

The IKEv1 daemon, `in.iked`, negotiates keys and authenticates IPsec SAs in a secure manner. IKEv1 provides perfect forward secrecy (PFS). In PFS, the keys that protect data transmission are not used to derive additional keys. Also, seeds used to create data transmission keys are not reused. See the [`in.iked\(1M\)`](#) man page.

IKEv1 Phase 1 Exchange

The IKEv1 protocol has two phases. Oracle Solaris supports the *Main Mode* Phase 1 exchange. The Main Mode exchange negotiates acceptable parameters to create an ISAKMP security

association (SA) between the two peers. This ISAKMP SA uses asymmetrical encryption to exchange its keying material and authenticates its peer using a preshared key or a public key certificate. Unlike IPsec SAs, the ISAKMP SAs are bidirectional, so only one security association is needed.

How IKEv1 negotiates ISAKAMP SAs in the Phase 1 exchange is configurable. IKEv1 reads the configuration information from the `/etc/inet/ike/config` file. Configuration information includes the following:

- Global parameters, such as the names of public key certificates
- Whether perfect forward secrecy (PFS) is required
- This system's IKE peers
- The algorithms that protect Phase 1 exchanges
- The authentication method

The two authentication methods are preshared keys and public key certificates. The public key certificates can be self-signed or issued by a [certificate authority \(CA\)](#).

For more information, see the [ike.config\(4\)](#) man page.

IKEv1 Phase 2 Exchange

The Phase 2 exchange is known as *Quick Mode*. The Quick Mode exchange negotiates the IPsec algorithms and keying material that is needed to create IPsec SAs. This exchange is protected (encrypted) by the ISAKMP SA that is negotiated in Phase 1.

The algorithms and security protocols in the Quick Mode exchange come from the IPsec policy file, `/etc/inet/ipsecinit.conf`.

The IPsec SAs are rekeyed when they expire. The lifetime of the SA is set by the `in.iked` daemon when it creates the IPsec SA. This value is configurable.

For more information, see the [ipseconf\(1M\)](#) and [in.iked\(1M\)](#) man pages.

IKEv1 Configuration Choices

The `/etc/inet/ike/config` configuration file contains the configuration for the `in.iked` daemon. The configuration consists of a number of rules. Each entry contains parameters such as algorithms and authentication data that this system can use with a similarly configured IKEv1 peer. The `in.iked` daemon supports preshared keys and public key certificates for identity.

The entry `auth_method preshared` indicates that preshared keys are used. Values for `auth_method` other than `preshared` indicate that public key certificates are to be used.

In IKEv1, preshared keys are tied to a particular IP address or range of addresses. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system.

For more information, see [“How IKE Works” on page 126](#) and the [ike.config\(4\)](#) and [ike.preshared\(4\)](#) man pages.

Configuring IKEv2

This chapter describes how to configure the Internet Key Exchange version 2 (IKEv2) for your systems. After IKEv2 is configured and enabled, it automatically generates keying material for the IPsec endpoints that it specifies. This chapter contains the following information:

- “Configuring IKEv2” on page 135
- “Configuring IKEv2 With Preshared Keys” on page 136
- “Initializing the Keystore to Store Public Key Certificates for IKEv2” on page 141
- “Configuring IKEv2 With Preshared Keys” on page 136

For overview information about IKE, see [Chapter 8, “About Internet Key Exchange”](#). For reference information about IKE, see [Chapter 12, “IPsec and Key Management Reference”](#). For more procedures, see the examples in the [ikeadm\(1M\)](#), [pktool\(1\)](#), [ikev2cert\(1M\)](#), [ikev2.config\(4\)](#), [in.ikev2d\(1M\)](#), and [kmfcfg\(1\)](#) man pages.

Configuring IKEv2

You can use preshared keys, self-signed certificates, and certificates from a certificate authority (CA) to authenticate IKE. Rules link a particular authentication method with the end points that are being protected. Therefore, you can use one or all authentication methods on a system. You can also run IKEv1 on an IKEv2 system. Typically, you run IKEv1 to protect communications with systems that do not support IKEv2. IKEv2 can also use a PKCS #11 hardware token for key and certificate storage.

Note - These tasks assume that the systems are assigned static IP addresses and you are running the network configuration profile `DefaultFixed`. If the `netadm list` command returns `Automatic`, see the [netcfg\(1M\)](#) man page for more information.

After configuring IKEv2, complete the IPsec procedures in [Chapter 7, “Configuring IPsec”](#) that use these IKEv2 rules to manage their keys. The following sections focus on specific IKEv2 configurations.

Configuring IKEv2 With Preshared Keys

If you are configuring peer systems or subnets to use IKEv2, and you are the administrator of these subnets, using preshared keys can be a good choice. Preshared keys might also be used when testing. For more information, see [“IKE With Preshared Key Authentication” on page 127](#).

▼ How to Configure IKEv2 With Preshared Keys

Substitute the names of your systems for the names `enigma` and `partym` in this procedure. You configure both IKE endpoints.

Before You Begin You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. **On each system, edit the `/etc/inet/ike/ikev2.config` file.**
`# pfedit /etc/inet/ike/ikev2.config`
2. **In the file, create a rule that uses preshared keys.**

Note - You will create the keys in [Step 4](#).

The rules and global parameters in this file must manage the keys in the IPsec policy in the system's `ipsecinit.conf` file. The following IKEv2 configuration examples manage the keys of the `ipsecinit.conf` examples in [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

- a. **For example, modify the `ikev2.config` file on the `enigma` system:**

Note - This example shows two transforms in the global parameters section. A peer can be configured with either of these transforms. To require a particular transform, include that transform in the rule.

```
### ikev2.config file on enigma, 192.168.116.16
```



```

## Global parameters
# This default value will apply to all transforms that follow
#
ikesa_lifetime_secs 3600
#
# Global transform definitions. The algorithm choices are
# based on RFC 4921.
#
## Two transforms are acceptable to this system, Group 20 and Group 19.
## A peer can be configured with 19 or 20.
## To ensure that a particular peer uses a specific transform,
## include the transform in the rule.
##
# Group 20 is 384-bit ECP - Elliptic Curve over Prime
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
# Group 19 is 256-bit ECP
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
#
## The rule to communicate with partym
## Label must be unique
{ Label "enigma-partym"
  auth_method preshared
  local_addr 192.168.116.16
  remote_addr 192.168.13.213
}

```

b. Modify the `ikev2.config` file on the `partym` system:

```

## ikev2.config file on partym, 192.168.13.213
## Global Parameters
#
...
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
...
## The rule to communicate with enigma
## Label must be unique
{ Label "partym-enigma"
  auth_method preshared
  local_addr 192.168.13.213
  remote_addr 192.168.116.16
}

```

3. On each system, verify the syntax of the file.

```
# /usr/lib/inet/in.ikev2d -c
```

4. Put the preshared key in the `/etc/inet/ike/ikev2.preshared` file on each system.



Caution - This file has special permissions and is owned by `ikeuser`. Never delete or replace this file. Instead, use the `pfedit` command to edit its contents so that the file retains its original properties.

- a. **For example, on the `enigma` system, the `ikev2.preshared` file would appear similar to the following:**

```
# pfedit -s /etc/inet/ike/ikev2.preshared
## ikev2.preshared on enigma, 192.168.116.16
#...
## label must match the rule that uses this key
{ label "enigma-partym"
## The preshared key can also be represented in hex
## as in 0xf47cb0f432e14480951095f82b
  key "This is an ASCII Cqret phrAz, use str0ng p@ssword tekniques"
}
```

For information about the options to the `pfedit` command, see the [`pfedit\(1M\)` man page](#).

- b. **On the `partym` system, the `ikev2.preshared` file is similar except for its unique label:**

```
## ikev2.preshared on partym, 192.168.13.213
#...
## label must match the label of the rule that uses this key
{ label "partym-enigma"
## 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 IKEv2 service instance.**

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

When replacing the preshared key, edit the preshared key files on the peer systems and restart the `ikev2` service.

```
# svcadm restart ikev2
```

Example 9-1 Using Different Local and Remote IKEv2 Preshared Keys

In this example, the IKEv2 administrators create a preshared key per system, exchange them, and add each key to the preshared key file. The label of the preshared key entry matches the label in a rule in the `ikev2.config` file. Then, they restart the `in.ikev2d` daemons.

After receiving the other system's preshared key, the administrator edits the `ikev2.preshared` file. The file on `partym` is the following:

```
# pfedit -s /etc/inet/ike/ikev2.preshared
#...
{ label "partym-enigma"
## local and remote preshared keys
local_key "P-LongISH key Th@t m^st Be Ch*angEd \'reguLarLy)"
remote_key "E-CHaNge lEyeGhtB+lBs et KeeS b4 2Lo0o0o0o0ng"
}
```

Therefore, the `ikev2.preshared` keys file on `enigma` must be the following:

```
#...
{ label "enigma-partym"
## local and remote preshared keys
local_key "E-CHaNge lEyeGhtB+lBs et KeeS b4 2Lo0o0o0o0ng"
remote_key "P-LongISH key Th@t m^st Be Ch*angEd \'reguLarLy)"
}
```

The administrators restart the IKEv2 service instance on each system.

```
# svcadm restart ikev2
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

For more examples, see the `ikev2.config(4)` and `ikev2.preshared(4)` man pages.

▼ How to Add a New Peer When Using Preshared Keys in IKEv2

If you add IPsec policy entries to a working configuration between the same peers, you need to refresh the IPsec policy service. You do not need to reconfigure or restart IKE.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKEv2 configuration.

Before You Begin You have updated the `ipsecinit.conf` file and refreshed IPsec policy for the peer systems.

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and “[How to Remotely Administer ZFS With Secure Shell](#)” in “[Managing Secure Shell Access in Oracle Solaris 11.2](#)” for secure remote login instructions.

1. Create a rule for IKEv2 to manage the keys for the new system that is using IPsec.

a. For example, on the enigma system, add the following rule to the `/etc/inet/ike/ikev2.config` file:

```
# pfedit ikev2.config
## ikev2.config file on enigma, 192.168.116.16
...
## The rule to communicate with ada
## Label must be unique
{label "enigma-ada"
  auth_method preshared
  local_addr 192.168.116.16
  remote_addr 192.168.15.7
}
```

For information about the options to the `pfedit` command, see the [pfedit\(1M\)](#) man page.

b. On the ada system, add the following rule:

```
## ikev2.config file on ada, 192.168.15.7
...
## The rule to communicate with enigma
{label "ada-enigma"
  auth_method preshared
  local_addr 192.168.15.7
  remote_addr 192.168.116.16
}
```

2. (Optional) On each system, verify the syntax of the file.

```
# /usr/lib/inet/in.ikev2d -c -f /etc/inet/ike/ikev2.config
```

3. Create an IKEv2 preshared key for the peer systems.

a. On the enigma system, add the following information to the `/etc/inet/ike/ikev2.preshared` file:

```
# pfedit -s /etc/inet/ike/ikev2.preshared
## ikev2.preshared on enigma for the ada interface
...
## The rule to communicate with ada
## Label must match the label of the rule
{ label "enigma-ada"
  # enigma and ada's shared key
```

```
key "Twas brillig and the slivey toves did *s0mEthiNg* be CareFULL hEEEr"
}
```

For information about the options to the `pfedit` command, see the [pfedit\(1M\)](#) man page.

b. On the ada system, add the following information to the `ikev2.preshared` file:

```
# ikev2.preshared on ada for the enigma interface
#
{ label "ada-enigma"
  # ada and enigma's shared key
  key "Twas brillig and the slivey toves did *s0mEthiNg* be CareFULL hEEEr"
}
```

4. On each system, read the changes into the kernel.

- **If the service is enabled, refresh it.**

```
# svcadm refresh ikev2
```

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

```
# svcadm enable ikev2
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

Initializing the Keystore to Store Public Key Certificates for IKEv2

To use public certificates with IKEv2, you must create a PKCS #11 keystore. The most commonly used keystore uses `pkcs11_softtoken`, which is provided by the Cryptographic Framework feature of Oracle Solaris.

The `pkcs11_softtoken` keystore for IKEv2 is in a directory that is owned by a special user, `ikeuser`. The default directory is `/var/user/ikeuser`. The user ID `ikeuser` is delivered with the system, but you must create the keystore. When you create the keystore, you create a PIN for the keystore. The IKEv2 service requires this PIN to log in to the keystore.

The `pkcs11_softtoken` keystore holds the private keys, public keys, and public certificates that are used by IKEv2. These keys and certificates are managed with the `ikev2cert` command,

which is a wrapper for the `pktool` command. The wrapper ensures that all keys and certificate operations are applied to the `pkcs11_softtoken` keystore that is owned by `ikeuser`.

If you have not added the PIN as a property value of the `ikev2` service, the following message displays in the `/var/log/ikev2/in.ikev2d.log` file:

```
date: (n) No PKCS#11 token "pin" property defined
for the smf(5) service: ike:ikev2
```

If you are not using public key certificates, you can ignore this message.

▼ How to Create and Use a Keystore for IKEv2 Public Key Certificates

You must create a keystore if you plan to use public certificates with IKEv2. To use the keystore, you must log in to it. When the `in.ikev2d` daemon starts, you or an automatic process supplies the PIN to the daemon. If site security permits automatic login, you must configure it. The default is an interactive login to use the keystore.

Before You Begin You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

1. Set the PIN for the IKEv2 keystore.

You use the `ikev2cert setpin` command to create the IKEv2 keystore. This command sets the owner of the PKCS #11 keystore to be `ikeuser`.

Do not use spaces in the PIN. For example, the value `WhatShouldIWrite` is valid, but the value `"What Should"` is not.

```
% pfbash
# /usr/sbin/ikev2cert setpin
Enter token passphrase: changeme
Create new passphrase:      Type strong passphrase
Re-enter new passphrase: xxxxxxxx
Passphrase changed.
```



Caution - Store this passphrase in a safe location. You need it to use the keystore.

2. Log in to the keystore automatically or interactively.

Automatic login is preferred. If site security policy does not permit automatic login, you must interactively log in to the keystore when the `in.ikev2d` daemon is restarted.

- **Configure the keystore to enable automatic login.**

- a. **Add the PIN as the value for the `pkcs11_softtoken/pin` service property.**

```
# svccfg -s ike:ikev2 editprop
```

A temporary edit window opens.

- b. **Uncomment the `setprop pkcs11_token/pin = line`.**

```
# setprop pkcs11_token/pin = astring: ()      Original entry
setprop pkcs11_token/pin = astring: ()      Uncommented entry
```

- c. **Replace the parentheses with the PIN from [Step 1](#).**

```
setprop pkcs11_token/pin = astring: PIN-from-Step-1
```

Leave a space between the colon and the PIN.

- d. **Uncomment the `refresh` line at the bottom of the file, then save your changes.**

```
# refresh
refresh
```

- e. **(Optional) Verify the value of the `pkcs11_token/pin` property.**

The `pkcs11_token/pin` property holds the value that is checked when accessing the keystore owned by `ikeuser`.

```
# svccfg -s ike:ikev2 listprop pkcs11_token/pin
pkcs11_token/pin    astring    PIN
```

- **When automatic keystore login is not configured, log in to the keystore manually.**

Run this command each time the `in.ikev2d` daemon starts.

```
# pfbash
# ikeadm -v2 token login "Sun Metaslot"
Enter PIN for PKCS#11 token 'Sun Metaslot':      Type the PIN from Step 1
ikeadm: PKCS#11 operation successful
```

3. (Optional) Verify that a PIN has been set in the keystore.

```
# ikev2cert tokens
Flags: L=Login required I=Initialized X=User PIN expired S=S0 PIN expired
Slot ID   Slot Name           Token Name           Flags
-----
1         Sun Crypto Softtoken Sun Software PKCS#11 softtoken LI
```

The LI in the Flags column indicates that the PIN is set.

4. To manually log out of the pkcs11_softtoken, use the ikeadm command.

```
# ikeadm -v2 token logout "Sun MetaSlot"
ikeadm: PKCS#11 operation successful
```

You might log out to limit communication between two sites to a finite period of time. By logging out, the private key becomes unavailable, so new IKEv2 sessions cannot be initiated. The existing IKEv2 session continues unless you delete the session keys with the `ikeadm delete ikesa` command. Preshared key rules continue to work. See the [ikeadm\(1M\)](#) man page.

Configuring IKEv2 With Public Key Certificates

Public certificates can be a good choice for large deployments. For more information, see [“IKE With Public Key Certificates” on page 128](#).

Public key certificates are stored in a softtoken keystore by the Cryptographic Framework. On systems with attached hardware, the certificates can also be generated and stored in the hardware. For the procedure, see [“How to Generate and Store Public Key Certificates for IKEv2 in Hardware” on page 157](#).

For background information, see [“How IKE Works” on page 126](#).

The following task map lists procedures for creating public key certificates for IKEv2. The procedures include how to store the certificates in a hardware keystore if your system has an attached Sun Crypto Accelerator 6000 board.

TABLE 9-1 Configuring IKEv2 With Public Key Certificates Task Map

Task	Description	For Instructions
Create a keystore for certificates.	Initializes the PKCS #11 keystore where the certificates for IKEv2 are stored.	“Initializing the Keystore to Store Public Key Certificates for IKEv2” on page 141
Configure IKEv2 with self-signed public key certificates.	Creates a public key certificate signed by you. Exports the certificate to peers and imports the peers' certificates.	“How to Configure IKEv2 With Self-Signed Public Key Certificates” on page 145
Configure IKEv2 with a certificate from a CA.	Requires you to create a CSR and then import all returned certificates into the keystore. Then, verify and import the IKE peers' certificates.	“How to Configure IKEv2 With Certificates Signed by a CA” on page 151
Configure how revoked certificates are handled.	Determines if CRLs are used and OCSP servers are polled, including how to handle network delays.	“How to Set a Certificate Validation Policy in IKEv2” on page 153

Task	Description	For Instructions
Configure the storage of certificates in the keystore of attached hardware.	Locates the Sun Crypto Accelerator 6000 board and configures IKEv2 to use it.	“How to Generate and Store Public Key Certificates for IKEv2 in Hardware” on page 157

▼ How to Configure IKEv2 With Self-Signed Public Key Certificates

In this procedure, you create and sign a public key certificate. The private key and certificate are stored in the PKCS #11 softtoken keystore for IKEv2. You send the public key certificate to IKE peers, who in turn, send you their public certificate.

You perform this procedure on all IKE systems that use self-signed certificates.

Before You Begin To use the certificates, you must have completed [“How to Create and Use a Keystore for IKEv2 Public Key Certificates” on page 142](#).

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be using a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. Create a self-signed certificate in the keystore.

For a description of the arguments to the `ikev2cert gencertcommand`, review the `pktool gencert keystore=pkcs11` subcommand in the [pktool\(1\)](#) man page.

For the format of the subject argument, see [“Using Public Key Certificates in IKE” on page 128](#).

Note - Give the certificate a label. The label identifies the certificate and its corresponding keys in the local keystore.

a. For example, the command on the `partym` system would appear similar to the following:

```
# pfbash
# ikev2cert gencert \
  label="ITpartym" \
  subject="O=exampleco, OU=IT, C=US, CN=partym" \
```

```
serial=0x87654321
keytype=rsa
keylen=2048
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

The following error messages indicate a mistyped PIN or an uninitialized keystore:

```
Error creating certificate and keypair:
keystore error: CKR_PIN_INCORRECT
libkmf error: KMF_ERR_AUTH_FAILED

Error creating certificate and keypair:
keystore error: CKR_PIN_EXPIRED: PIN expired and must be changed
libkmf error: KMF_ERR_BAD_PARAMETER: invalid parameter
```

Tip - A display of `pktool` command syntax indicates that part of your certificate entry is mistyped. Examine the command for the use of a disallowed algorithm, missing double quotes and equals signs, and other typographical errors. One strategy to locate the invalid argument is to retrieve the command, then remove the arguments one at a time.

b. The command on the `enigma` system would appear similar to the following:

```
# ikev2cert gencert \
label=ITenigma \
subject="O=exampleco, OU=IT, C=US, CN=enigma" \
serial=0x86428642
keytype=rsa
keylen=2048
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

2. (Optional) List the keys and certificate.

```
enigma # /usr/sbin/ikev2cert list objtype=both
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
No.    Key Type    Key Len.    Key Label
-----
Asymmetric private keys:
1)    RSA                ITenigma
Asymmetric public keys:
1)    RSA                ITenigma
Certificates:
1) X.509 certificate
Label: ITenigma
Subject: C=US, O=exampleco, OU=IT, CN=enigma
Issuer: C=US, O=exampleco, OU=IT, CN=enigma
Not Before: April 10 21:49:00 2014 GMT
Not After: April 10 21:49:00 2015 GMT
Serial: 0x86426420
Signature Algorithm: sha1WithRSAEncryption
X509v3 Subject Key Identifier:
    34:7a:3b:36:c7:7d:4f:60:ed:ec:4a:96:33:67:f2:ac:87:ce:35:cc
SHA1 Certificate Fingerprint:
```

```
68:07:48:65:a2:4a:bf:18:f5:5b:95:a5:01:42:c0:26:e3:3b:a5:30
```

Tip - The default hash algorithm is SHA1. To create a certificate with a stronger signature algorithm, use the `keytype` option and a different hash algorithm, such as `keytype=rsa hash=sha384`. For the options, see the [pktool\(1\)](#) man page.

3. Deliver the certificate to the other system.

a. On each system, export only the certificate to a file.

The `outformat=pem` option ensures that the public certificate is placed in the file in a format that is suitable for direct import. The label identifies the certificate in the keystore.

```
# cd /tmp
# ikev2cert export objtype=cert outformat=pem outfile=filename label=label
Enter PIN for Sun Software PKCS#11 softtoken:xxxxxxx
```

b. Send the certificate to the other system by email, sftp, or ssh.

For example, if you administer both systems, use the `sftp` command to bring over the certificate from the other system.

```
enigma # sftp jdoe@partym:/tmp/ITpartym.pem /tmp/ITpartym.pem.cert
partym # sftp jdoe@enigma:/tmp/ITenigma.pem /tmp/ITenigma.pem.cert
```

You are prompted for your password. In this example, `jdoe` must provide a password.

4. Verify that the certificates are identical.

You want to ensure that you have received the proper certificate *before* you load it into your keystore.

a. Create a digest of the exported file on each system.

For example, the `partym` administrator emails the digest of the file that contains `partym`'s certificate to the other administrator. The `enigma` administrator emails the digest of the `enigma` certificate file.

```
partym # digest -a sha1 /tmp/ITpartym.pem
c6dbef4136c0141ae62110246f288e5546a59d86

enigma # digest -a sha1 ITenigma.pem
6b288a6a6129d53a45057065bd02b35d7d299b3a
```

b. On the other system, run the digest command on the file that contains the certificate from the first system.

```
enigma # digest -a sha1 /tmp/ITpartym.pem.cert
c6dbef4136c0141ae62110246f288e5546a59d86
```

```
partym # digest -a sha1 /tmp/ITenigma.pem.cert
6b288a6a6129d53a45057065bd02b35d7d299b3a
```

The digests must match. If they do not match, do not import the file to the keystore. For another way of verifying certificate validity, see [Example 9-3](#).

5. After verification, import the other system's certificate to your keystore.

When you import the certificate into your keystore, you must assign a label to it that uniquely identifies the certificate on your system. The label links the public key with its public key certificate.

```
enigma# ikev2cert import label=ITpartym1 infile=/tmp/ITpartym.pem.cert
```

```
partym# ikev2cert import label=ITenigma1 infile=/tmp/ITenigma.pem.cert
```

6. (Optional) List the objects in your keystore.

Compare the listing with the list in [Step 2](#). For example, in the enigma keystore, the partym certificate is added.

```
enigma # /usr/sbin/ikev2cert list objtype=both
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
No.      Key Type      Key Len.      Key Label
-----
Asymmetric private keys:
1)      RSA              ITenigma
Asymmetric public keys:
1)      RSA              ITenigma
Certificates:
1) X.509 certificate
Label: ITenigma
Subject: C=US, O=exampleco, OU=IT, CN=enigma
Issuer: C=US, O=exampleco, OU=IT, CN=enigma
Not Before: April 10 21:49:00 2014 GMT
Not After: April 10 21:49:00 2015 GMT
Serial: 0x86426420
Signature Algorithm: sha1WithRSAEncryption
X509v3 Subject Key Identifier:
34:7a:3b:36:c7:7d:4f:60:ed:ec:4a:96:33:67:f2:ac:87:ce:35:cc
SHA1 Certificate Fingerprint:
68:07:48:65:a2:4a:bf:18:f5:5b:95:a5:01:42:c0:26:e3:3b:a5:30

2) X.509 certificate
Label: ITpartym1
Subject: C=US, O=exampleco, OU=IT, CN=partym
Issuer: C=US, O=exampleco, OU=IT, CN=partym
Not Before: April 10 21:40:00 2014 GMT
Not After: April 10 21:40:00 2015 GMT
Serial: 0x87654321
Signature Algorithm: sha1WithRSAEncryption
X509v3 Subject Key Identifier:
```

```

ae:d9:c8:a4:19:68:fe:2d:6c:c2:9a:b6:06:55:b5:b5:d9:d9:45:c6
SHA1 Certificate Fingerprint:
83:26:44:29:b4:1f:af:4a:69:0d:87:c2:dc:f4:a5:1b:4f:0d:36:3b

```

7. On each system, use the certificates in an IKEv2 rule.

Use the `pfedit` command to edit the `/etc/inet/ike/ikev2.config` file.

a. For example, on the `partym` system, the rule in the `ikev2.config` file would appear similar to the following:

```

## ... Global transform that applies to any rule without a declared transform
ikesa_xform { dh_group 21 auth_alg sha512 encr_alg aes }
## ... Any self-signed
## end-entity certificates must be present in the keystore or
## they will not be trusted.
{
  label "partym-enigma"
  auth_method cert
  local_id DN = "O=exampleco, OU=IT, C=US, CN=partym"
  remote_id DN = "O=exampleco, OU=IT, C=US, CN=enigma"
}
...

```

b. On the `enigma` system, use the DN of the `enigma` certificate for the value of `local_id` in the `ikev2.config` file.

For the remote parameter, use the `partym` certificate's DN as the value. Ensure that the value for the `label` keyword is unique on the local system.

```

...
ikesa_xform { dh_group 21 auth_alg sha512 encr_alg aes }
...
{
  label "enigma-partym"
  auth_method cert
  local_id DN = "O=exampleco, OU=IT, C=US, CN=enigma"
  remote_id DN = "O=exampleco, OU=IT, C=US, CN=partym"
}
...

```

8. (Optional) On each system, check the validity of the `ikev2.config` files.

```
# /usr/lib/inet/inikev2.d -c
```

Fix any typographical errors or inaccuracies before continuing.

9. On each system, check the state of the IKEv2 service instance.

```

# svcs ikev2
STATE      STIME      FMRI
disabled   Sep_07     svc:/network/ipsec/ike:ikev2

```

10. On each system, enable the IKEv2 service instance.

```
partym # svcadm enable ipsec/ike:ikev2
```

```
enigma # svcadm enable ipsec/ike:ikev2
```

Example 9-2 Creating a Self-Signed Certificate With a Limited Lifetime

In this example, the administrator specifies that the certificate is valid for two years.

```
# ikev2cert gencert \  
> label=DBAuditV \  
> serial=0x12893467235412 \  
> subject="O=exampleco, OU=DB, C=US, CN=AuditVault" \  
> altname=EMAIL=auditV@example.com \  
> keytype=ec curve=secp521r1 hash=sha512 \  
> lifetime=2-year
```

Example 9-3 Verifying a Public Key Certificate by Its Fingerprint

In this example, the administrator uses the certificate fingerprint to verify the certificate. The disadvantage of this method is that the administrator must import the peer's certificate into the keystore before viewing the fingerprint.

The administrator imports the certificate, lists it with the `ikev2cert list objtype=cert` command, then copies the certificate fingerprint from the output and sends it to the other system administrator.

```
SHA1 Certificate Fingerprint:  
83:26:44:29:b4:1f:af:4a:69:0d:87:c2:dc:f4:a5:1b:4f:0d:36:3b
```

If the verification fails, the administrator who imported the certificate must remove it and its public key from the keystore.

```
# ikev2cert delete label=label-name  
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx  
1 public key(s) found, do you want to delete them (y/N) ? y  
1 certificate(s) found, do you want to delete them (y/N) ? y
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Configure IKEv2 With Certificates Signed by a CA

Organizations that protect a large number of communicating systems typically use public certificates from a certificate authority (CA). For background information, see [“IKE With Public Key Certificates” on page 128](#).

You perform this procedure on all IKE systems that use certificates from a CA.

Before You Begin To use the certificates, you must have completed [“How to Create and Use a Keystore for IKEv2 Public Key Certificates” on page 142](#).

You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. Change to a writable directory.

The following error message can indicate that the CSR file cannot be written to disk:

```
Warning: error accessing "CSR-file"
```

For example, use the /tmp directory.

```
# cd /tmp
```

2. Create a certificate signing request.

You use the `ikev2cert gencsr` command to create a certificate signing request (CSR). For a description of the arguments to the command, review the `pktool gencsr keystore=pkcs11` subcommand in the [pktool\(1\)](#) man page.

For example, the following command creates a file that contains the CSR on the `partym` system:

```
# pfbash
# /usr/sbin/ikev2cert gencsr \
keytype=rsa
keylen=2048
label=Partym1 \
outcsr=/tmp/Partymcsr1 \
subject="C=US, O=PartyCompany\, Inc., OU=US-Partym, CN=Partym"
Enter PIN for Sun Software PKCS#11 softtoken: xxxxxxxx
```

3. (Optional) Copy the contents of the CSR for pasting into the CA's web form.

```
# cat /tmp/Partymcsr1
-----BEGIN CERTIFICATE REQUEST-----
MIICKDCCAoCAQAwTzELMAkGA1UEBhMCMVVMxGZAZBgNVBAoTElBhcnR5Q29tcGFu
eSwgSW5jLjESMBAGA1UECXMJVVMtUGFydHltMQ8wDQYDVQQDEWZQYXJ0eW0wggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCMBINmgZ4XWUv2q1fshZUN/SLb
WNLXZxdKwt5e71o0owjyby69eL7HE0QBUij73nTkXE3n4gxobjZE+hvJ6GOCbREA
jgSquP2US7Bn9XEcXRrsOc7MCFPrsA+hVICNHpKNseUOU/rg+wzoo5hA1ixtWuXH
bYDeEWQI5tLZgDZoCWGrdHEjwVvHfVz+a0WBjyZBY0ueBhXaa68QqS0SnRVDX56Q
3p4H/AR4h0dcSja72XmMKPU5p3RVb8n/hrfKjIdjiGYXD4D+WZxQ65xxCcnALvVH
nZHU\AtP7QHx4RXlQVNNwEsY6C95RX9297rNWLsYvp/86xWrQkTlNqVAeUkHAgMB
AAEWcWYJKoZIHvcNAQEFA4IBAQB3R6rmZdqcgN8Tomyjp2CFTdyAWixkIATXpLM1
GL5ghrnDvadD61M+vS1yhFLIcSNM8fLRrCHIKtAmB8ITnggJ//rzbHq3jdlA/iQt
kgGoTXfz8j6B57Ud6l+MBLiBSBy0QK4GIg80jlb9Kk5HsZ48mIoI/Qb7FFW4p9dB
JEUh0eYhkaGtwJ21YNNvKg0e0cnSZy+xP9Wa9WpfdSBO4TicLDw0Yq7koNnfL0IB
Fj2bt/wI7iZ1DcpwphsiwnW9K9YynAJZzHd1ULVpn5Kd7vSRz9youLLzSb+9i1g0
E43Dw0hRk6P/Uq0N4e1Zca4otezNxyEqLPZI7pJ5u0o0sbiw
-----END CERTIFICATE REQUEST-----
```

4. Submit the CSR to a [certificate authority \(CA\)](#).

The CA can tell you how to submit the CSR. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your CSR into the form.

Tip - Some web forms have an Advanced button where you can paste your certificate. The CSR is generated in PKCS#10 format. Therefore, find the portion of the web form that mentions PKCS#10.

5. Import each certificate that you receive from the CA into your keystore.

The `ikev2cert import` imports the certificate into the keystore.

a. Import the public key and certificate that you received from the CA.

```
# ikev2cert import objtype=cert label=Partym1 infile=/tmp/Partym1Cert
```

Tip - For administrative convenience, assign the same label to the imported certificate as the label of the original CSR.

b. Import the root certificate from the CA.

```
# ikev2cert import objtype=cert infile=/tmp/Partym1CAcert
```

c. Import any intermediate CA certificates into the keystore.

Tip - For administrative convenience, assign the same label to the imported intermediate certificates as the label of the original CSR.

If the CA has sent separate files for each intermediate certificate, then import them as you imported the preceding certificates. However, if the CA delivers its certificate chain as a PKCS#7 file, you must extract the individual certificates from the file, then import each certificate as you imported the preceding certificates:

Note - You must assume the root role to run the `openssl` command. See the [openssl\(5\)](#) man page.

```
# openssl pkcs7 -in pkcs7-file -print_certs
# ikev2cert import objtype=cert label=Partym1 infile=individual-cert
```

6. Set the certificate validation policy.

If the certificate contains sections for CRLs or OCSP, you must configure the certificate validation policy according to your site requirements. For instructions, see [“How to Set a Certificate Validation Policy in IKEv2” on page 153](#).

7. After you complete the procedure on all IKE systems which use your certificate, enable the ikev2 service on all systems.

The peer systems need the [trust anchor](#) certificate and a configured `ikev2.config` file.

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Set a Certificate Validation Policy in IKEv2

You can configure several aspects of how certificates are handled for your IKEv2 system.

Before You Begin You must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. Review the default certificate validation policy.

Certificate policy is set at installation in the `/etc/inet/ike/kmf-policy.xml` file. The file is owned by `ikeuser` and is modified by using the `kmfcfg` command. The default certificate validation policy is to download CRLs to the `/var/user/ikeuser/crls` directory. The use of OCSP is also enabled by default. If your site requires a proxy to reach the Internet, you must configure the proxy. See [“How to Handle Revoked Certificates in IKEv2”](#) on page 155.

```
# pfbash
# kmfcfg list dbfile=/etc/inet/ike/kmf-policy.xml policy=default
Policy Name: default
Ignore Certificate Validity Dates: false      Unknown purposes or applications for the certificate
Ignore Unknown EKUs: false
Ignore Trust Anchor in Certificate Validation: false
Trust Intermediate CAs as trust anchors: false
Maximum Certificate Path Length: 32
Certificate Validity Period Adjusted Time leeway: [not set]
Trust Anchor Certificate: Search by Issuer
Key Usage Bits: 0      Identifies critical parts of certificate
Extended Key Usage Values: [not set]      Purposes or applications for the certificate
HTTP Proxy (Global Scope): [not set]
Validation Policy Information:
  Maximum Certificate Revocation Responder Timeout: 10
  Ignore Certificate Revocation Responder Timeout: true
  OCSP:
    Responder URI: [not set]
    OCSP specific proxy override: [not set]
    Use ResponderURI from Certificate: true
    Response lifetime: [not set]
    Ignore Response signature: false
    Responder Certificate: [not set]
  CRL:
    Base filename: [not set]
    Directory: /var/user/ikeuser/crls
    Download and cache CRL: true
    CRL specific proxy override: [not set]
    Ignore CRL signature: false
    Ignore CRL validity date: false
IPsec policy bypass on outgoing connections: true
Certificate to name mapper name: [not set]
Certificate to name mapper pathname: [not set]
Certificate to name mapper directory: [not set]
Certificate to name mapper options: [not set]
```

2. Review the certificate for features that indicate the validation options to modify.

For example, a certificate that includes a CRL or OCSP URI can use a validation policy that specifies the URI to use to check certificate revocation status. You might also configure timeouts.

3. Review the [kmfcfg\(1\)](#) man page for configurable options.

4. Configure the certificate validation policy.

For a sample policy, see [“How to Handle Revoked Certificates in IKEv2” on page 155](#).

▼ How to Handle Revoked Certificates in IKEv2

Revoked certificates are certificates that are compromised for some reason. A revoked certificate that is in use is a security risk. You have options when verifying certificate revocation. You can use a static list or you can verify revocations dynamically over the HTTP protocol.

Before You Begin You have received and installed certificates from a CA.

You are familiar with the CRL and OSCP methods of checking for revoked certificates. For information and pointers, see [“IKE With Public Key Certificates” on page 128](#).

You must become an administrator who is assigned the Network IPsec Management rights profile, and use a profile shell. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

1. Locate the CRL and OSCP sections in the certificate that you received from the CA.

You can identify the certificate from the label of your CSR.

```
# pfbash
# ikev2cert list objtype=cert | grep Label:
Enter PIN for Sun Software PKCS#11 softtoken:
Label: Partym1
```

For example, the following truncated output highlights the CRL and OSCP URIs in a certificate.

```
# ikev2cert list objtype=cert label=Partym1
X509v3 extensions:
...
X509v3 CRL Distribution Points:
  Full Name:
    URI:http://onsitecrl.PKI.example.com/OCCIPsec/LatestCRL.crl
X509v3 Authority Key Identifier:
...
Authority Information Access:
  OCSP - URI:http://ocsp.PKI.example.com/revokes/
X509v3 Certificate Policies:
  Policy: 2.16.840.1.113733.1.7.23.2
```

Under the CRL Distribution Points entry, the URI value indicates that this organization's CRL is available in a file on the web. The OSCP entry indicates that the status of individual certificates can be determined dynamically from a server.

2. Enable the use of CRLs or an OCSP server by specifying a proxy.

```
# kmfcfg modify \  
dbfile=/etc/inet/ike/kmf-policy.xml \  
policy=default \  
http-proxy=www-proxy.ja.example.com:80
```

At sites where a proxy is optional, you do not need to specify one.

3. Verify that the certificate validation policy is updated.

For example, verify that the OCSP was updated.

```
# kmfcfg list \  
dbfile=/etc/inet/ike/kmf-policy.xml \  
policy=default  
...  
OCSP:  
  Responder URI: [not set]  
  Proxy: www-proxy.ja.example.com:80  
  Use ResponderURI from Certificate: true  
  Response lifetime: [not set]  
  Ignore Response signature: false  
  Responder Certificate: [not set]
```

4. Restart the IKEv2 service.

```
# svcadm restart ikev2
```

5. (Optional) Stop using CRLs or OCSP.

■ **To stop using CRLs, type:**

```
# pfexec kmfcfg modify \  
dbfile=/etc/inet/ike/kmf-policy.xml policy=default \  
crl-none=true
```

The `crl-none=true` argument forces the system to use downloaded CRLs from the local cache.

■ **To stop using OCSP, type:**

```
# pfexec kmfcfg modify \  
dbfile=/etc/inet/ike/kmf-policy.xml policy=default \  
ocsp-none=true
```

Example 9-4 Changing the Time That a System Waits For IKEv2 Certificate Verification

In this example, the administrator limits the wait to twenty seconds for a certificate to be verified.

```
# kmfcfg modify dbfile=/etc/inet/ike/kmf-policy.xml policy=default \  
wait-time=20
```

```
cert-revoke-responder-timeout=20
```

By default, when a response times out, the authentication of the peer succeeds. Here, the administrator configures a policy where the connection is refused when authentication fails. In this configuration, certificate validation fails when an OCSP or CRL server becomes unresponsive.

```
# kmfcfg modify dbfile=/etc/inet/ike/kmf-policy.xml policy=default \
  ignore-cert-revoke-responder-timeout=false
```

To activate the policy, the administrator restarts the IKEv2 service.

```
# svcadm restart ikev2
```

▼ How to Generate and Store Public Key Certificates for IKEv2 in 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.

Generating and storing public key certificates on hardware is similar to generating and storing public key certificates on your system. On hardware, the `ikev2cert gencert token=hw-keystore` command is used to identify the hardware keystore.

Before You Begin This 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 hardware keystore has been configured. For instructions, see the [Sun Crypto Accelerator 6000 Board Product Library Documentation \(http://docs.oracle.com/cd/E19321-01/index.html\)](http://docs.oracle.com/cd/E19321-01/index.html). These instructions include setting up the keystore.

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”.

If you administer remotely, see [Example 7-1](#) and “How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2” for secure remote login instructions.

1. **Confirm that you have a token ID for the attached Sun Crypto Accelerator 6000 board.**

```
# pfbash
# ikev2cert tokens
```

```
Flags: L=Login required I=Initialized X=User PIN expired S=SO PIN expired
Slot ID Slot Name Token Name Flags
```

```

-----
1      sca6000                                sca6000                                LI
2      n2cp/0 Crypto Accel Bulk 1.0          n2cp/0 Crypto Accel Bulk 1.0
3      ncp/0 Crypto Accel Asym 1.0          ncp/0 Crypto Accel Asym 1.0
4      n2rng/0 SUNW_N2_Random_Number_Ge     n2rng/0 SUNW_N2_RNG
5      Sun Crypto Softtoken                  Sun Software PKCS#11 softtoken    LI

```

2. Generate a self-signed certificate or a CSR, and specify the token ID.

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.

Choose one of the following options:

■ **For a self-signed certificate, use this syntax:**

```

# ikev2cert gencert token=sca6000 keytype=rsa \
hash=sha256 keylen=2048 \
subject="CN=FortKnox, C=US" serial=0x6278281232 label=goLdrepo
Enter PIN for sca6000:      See Step 3

```

■ **For a certificate signing request, use this syntax:**

```

# ikev2cert gencsr token=sca6000 -i
> keytype=
> hash=
> keylen=
> subject=
> serial=
> label=
> outcsr=
Enter PIN for sca6000 token:      See Step 3

```

For a description of the arguments to the `ikev2cert` command, see the [pktool\(1\)](#) man page.

3. At the prompt for a PIN, type the Sun Crypto Accelerator 6000 user name, a colon, and the user's password.

Note - You must know the user name and the password for the keystore.

If the Sun Crypto Accelerator 6000 board is configured with a user `admin` whose password is `inThe%4ov`, you would type the following:

```

Enter PIN for sca6000 token: admin:inThe%4ov
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCMVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...

```

```
oKUDbbZ90/pLWYGr
-----END X509 CERTIFICATE-----
```

4. 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 message.

- **Send the certificate signing request to the CA.**

Follow the instructions of the CA to submit the CSR. For a more detailed discussion, see [“Using Public Key Certificates in IKE” on page 128](#).

5. Import the certificates into the hardware keystore.

Import the certificates that you received from the CA and provide the user and PIN from [Step 3](#).

```
# ikev2cert import token=sca6000 infile=/tmp/DCA.ACCEL.CERT1
Enter PIN for sca6000 token:      Type user:password
# ikev2cert import token=sca6000 infile=/tmp/DCA.ACCEL.CA.CERT
Enter PIN for sca6000 token:      Type user:password
```

6. Enable the hardware keystore to be used automatically or interactively.

Automatic login is preferred. If site security policy does not permit automatic login, you must interactively log in to the keystore when the `in.ikev2d` daemon is restarted.

- **Configure automatic login to the keystore.**

- a. **Add the PIN as the value for the `pkcs11_token/uri` service property.**

For a description of this property, see [“IKEv2 Service” on page 217](#).

```
# svccfg -s ike:ikev2 editprop
```

A temporary edit window opens.

- b. **Uncomment the `setprop pkcs11_token/uri =` line and replace the parentheses with the name of the token in the following format:**

```
# setprop pkcs11_token/uri = ()      Original entry
setprop pkcs11_token/uri = pkcs11:token=sca6000
```

- c. **Uncomment the `setprop pkcs11_token/uri =` line and replace the parentheses with the `username:PIN` from [Step 3](#).**

```
# setprop pkcs11_token/uri = ()      Original entry
setprop pkcs11_token/uri = admin:PIN-from-Step-3
```

- d. **Uncomment the refresh line at the bottom of the file, then save your changes.**

```
# refresh
refresh
```

- e. **(Optional) Verify the value of the pkcs11_token properties.**

```
# svccfg -s ikev2 listprop pkcs11_token
pkcs11_token/pin    astring    username:PIN
pkcs11_token/uri    astring    pkcs11:token=sca6000
```

- **If automatic login is not configured, log in to the hardware keystore manually.**

Run this command each time the `in.ikey2d` daemon starts.

```
# pfexec ikeadm -v2 token login sca6000
Enter PIN for sca6000 token: admin:PIN-from-Step-3
ikeadm: sca6000 operation successful
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

Configuring IKEv1

This chapter describes how to configure the Internet Key Exchange version 1 (IKEv1) for your systems. After IKEv1 is configured, it automatically generates keying material for IPsec on your network. This chapter contains the following information:

- “Configuring IKEv1 With Preshared Keys” on page 162
- “Configuring IKEv1 With Public Key Certificates” on page 166
- “Configuring IKEv1 for Mobile Systems” on page 184
- “Configuring IKEv1 to Find Attached Hardware” on page 191

Note - If you plan to implement IKEv2 only, proceed to [Chapter 9, “Configuring IKEv2”](#).

For overview information about IKE, see [Chapter 8, “About Internet Key Exchange”](#). For reference information about IKE, see [Chapter 12, “IPsec and Key Management Reference”](#). For more procedures, see the Examples sections of the [ikeadm\(1M\)](#), [ikecert\(1M\)](#), and [ike.config\(4\)](#) man pages.

Note - These tasks assume that the systems are assigned static IP addresses and you are running the network configuration profile `DefaultFixed`. If the `netadm list` command returns `Automatic`, see the [netcfg\(1M\)](#) man page for more information.

Configuring IKEv1

You can use preshared keys, self-signed certificates, and certificates from a certificate authority (CA) to authenticate IKE. A rule in the `ike/config` file links the particular IKEv1 authentication method with the IKEv1 peer. Therefore, you can use one or all IKE authentication methods on a system. A pointer to a PKCS #11 library enables IKEv1 to use an attached hardware accelerator.

After configuring IKEv1, complete the IPsec task in [Chapter 7, “Configuring IPsec”](#) that uses the IKEv1 configuration.

Configuring IKEv1 With Preshared Keys

If you are configuring peer systems or subnets to use IKEv1 and you are the administrator of these subnets, using preshared keys can be a good choice. Preshared keys might also be used when testing. For more information, see [“IKE With Preshared Key Authentication”](#) on page 127.

▼ How to Configure IKEv1 With Preshared Keys

The IKE implementation offers algorithms whose keys vary in length. The key length that you choose is determined by site security. In general, longer keys provide more security than shorter keys.

In this procedure, you generate keys in ASCII format.

These procedures use the system names `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. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell”](#) in [“Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

- 1. On each system, create an `/etc/inet/ike/config` file.**

You can use the `/etc/inet/ike/config.sample` as a template.

- 2. Enter rules and global parameters in the `ike/config` file on each system.**

The rules and global parameters in this file should permit the IPsec policy in the system's `ipsecinit.conf` file to succeed. The following IKEv1 configuration examples work with the

ipseccinit.conf examples in [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

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. Put the preshared key in the /etc/inet/secret/ike.preshared file on each system.

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 IKEv1 service.

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

Example 10-1 Refreshing an IKEv1 Preshared Key

When IKEv1 administrators want to refresh the preshared key, they edit the files on the peer systems and restart the `in.iked` daemon.

First, on every system in the two subnets that uses the preshared key, the administrator changes the preshared key entry.

```
# pfedit -s /etc/inet/secret/ike.preshared
...
{ localidtype IP
  localid 192.168.116.0/24
  remoteidtype IP
  remoteid 192.168.13.0/24
  # The two subnet's shared passphrase for keying material
  key "L0ooong key Th@t m^st Be Ch*angEd \'reguLarLy)"
}
```

Then, the administrator restarts the IKEv1 service on every system.

For information about the options to the `pfedit` command, see the [pfedit\(1M\)](#) man page.

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

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Update IKEv1 for a New Peer System

If you add IPsec policy entries to a working configuration between the same peers, you need to refresh the IPsec policy service. You do not need to reconfigure or restart IKEv1.

If you add a new peer to the IPsec policy, in addition to the IPsec changes, you must modify the IKEv1 configuration.

Before You Begin You have updated the `ipseccinit.conf` file and refreshed IPsec policy for the peer systems.

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. **Create a rule for IKEv1 to manage the keys for the new system that is using IPsec.**
 - a. **For example, on the `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 IKEv1 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:default
```

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

Configuring IKEv1 With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Public certificates from a certificate authority (CA) typically require negotiation with an outside organization. The certificates very easily scale to protect a large number of communicating systems.

Public key certificates can also be generated and stored in attached hardware. For the procedure, see [“Configuring IKEv1 to Find Attached Hardware” on page 191](#).

All certificates have a unique name in the form of an X.509 [distinguished name \(DN\)](#). Additionally, a certificate might have one or more subject alternative names, such as an email address, a DNS name, an IP address, and so on. You can identify the certificate in the IKEv1 configuration by its full DN or by one of its subject alternative names. The format of these alternative names is *tag=value*, where the format of the value corresponds to its tag type. For example, the format of the email tag is *name@domain.suffix*.

The following task map lists procedures for creating public key certificates for IKEv1. The procedures include how to accelerate and store the certificates on attached hardware..

TABLE 10-1 Configuring IKEv1 With Public Key Certificates Task Map

Task	Description	For Instructions
Configure IKEv1 with self-signed public key certificates.	Creates and places keys and two certificates on each system: <ul style="list-style-type: none"> ■ A self-signed certificate and its keys ■ The public key certificate from the peer system 	“How to Configure IKEv1 With Self-Signed Public Key Certificates” on page 168
Configure IKEv1 with a certificate authority.	Creates a certificate signing request, and then places certificates from the CA on each system. See “Using Public Key Certificates in IKE” on page 128 .	“How to Configure IKEv1 With Certificates Signed by a CA” on page 173
Configure public key certificates in local hardware.	Involves one of: <ul style="list-style-type: none"> ■ Generating a self-signed certificate in the local hardware, then adding the public key from a remote system to the hardware. ■ Generating a certificate signing 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 for IKEv1 in Hardware” on page 178
Update the certificate revocation list (CRL) from the CA.	Accesses the CRL from a central distribution point.	“How to Handle Revoked Certificates in IKEv1” on page 181

Note - To label packets and IKE negotiations on a Trusted Extensions system, follow the procedures in [“Configuring Labeled IPsec” in “Trusted Extensions Configuration and Administration”](#).

Public key certificates are managed in the global zone on Trusted Extensions systems. Trusted Extensions does not change how certificates are managed and stored.

<code>-t keytype</code>	Specifies the type of algorithm to use.
<code>-D dname</code>	Specifies the X.509 distinguished name (DN) for the certificate subject. For an example, see “Using Public Key Certificates in IKE” on page 128 .
<code>-A altname</code>	Specifies the alternate name or nickname for the certificate. The <i>altname</i> is in the form of <i>tag=value</i> . Valid tags are IP, DNS, email, and DN.

Note - The values of the `-D` and `-A` options are names that identify the certificate only, not any system, such as 192.168.13.213. In fact, because these values are certificate nicknames, you must verify out of band that the correct certificate is installed on the peer systems.

b. The command on the enigma system would appear similar to the following:

```
# ikecert certlocal -ks -m 2048 -t rsa-sha512 \
-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-----
MIIC1TCCAb2gAwIBAgIEB15JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
...
y85m6LHJYtC6
-----END X509 CERTIFICATE-----
```

2. Save the certificate and send it to the remote system.

The output is an encoded version of the public portion of the certificate. You can safely paste this certificate into an email message. The receiving party must verify out of band that they installed the correct certificate, as shown in [Step 4](#).

a. For example, you would send the public portion of the partym certificate to the enigma administrator.

```
To: admin@enigma.ja.example.com
From: admin@party.us.example.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@party.us.example.com
From: admin@enigma.ja.example.com
Message: ----BEGIN X509 CERTIFICATE-----
MIIC1TCCAb2gAwIBAgIEBL5JnjANBgkqhkiG9w0BAQUFADAaMRgwFgYDVQQDEw9T
...
y85m6LHJYtC6
-----END X509 CERTIFICATE-----
```

3. On each system, add the certificate that you received to the public key database.

a. Save the administrator's email to a file that is read by root.

b. Redirect the file to the `ikecert` command.

```
# ikecert certdb -a < /tmp/certificate.eml
```

The command imports the text between the BEGIN and END tags.

4. Verify with the other administrator that the certificate is from that administrator.

For example, you can telephone the other administrator to verify that the hash of their public certificate, which you have, matches the hash of their private certificate, which only they have.

a. List the stored certificate on `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: FEA65C5387BBF3B2C8F16C019FEB388
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-sha512
Key Size: 2048
```

Public key hash: FEA65C5387BBF3B2C8F16C019FEB388

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: FEA65C5387BBF3B2C8F16C019FEB388

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-party to JA-enigma"
  local_id_type dn
  local_id "O=exampleco, OU=IT, C=US, CN=party"
  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 party values. Ensure that the value for the label keyword is unique on the local system.

```
...
{
  label "JA-enigma to US-party"
  local_id_type dn
  local_id "O=exampleco, OU=IT, C=US, CN=enigma"
  remote_id "O=exampleco, OU=IT, C=US, CN=party"

  local_addr 192.168.116.16
  remote_addr 0.0.0.0/0

  p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg sha256 encr_alg aes}
}
```

6. On the peer systems, enable IKEv1.

```
party # svcadm enable ipsec/ike:default
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. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN”](#)

With IPsec” on page 108. For other examples of IPsec policy, see “How to Secure Network Traffic Between Two Servers With IPsec” on page 102.

▼ How to Configure IKEv1 With Certificates Signed by a CA

Before You Begin You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”.

If you administer remotely, see Example 7-1 and “How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2” for secure remote login instructions.

1. **Use the `ikecert certlocal -kc` command to create a certificate signing request (CSR).**

For a description of the arguments to the command, see Step 1 in “How to Configure IKEv1 With Self-Signed Public Key Certificates” on page 168.

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname
```

- a. **For example, the following command creates a CSR on the `partym` system:**

```
# ikecert certlocal -kc -m 2048 -t rsa-sha384 \
> -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-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zck080m09X
-----END CERTIFICATE REQUEST-----
```

- b. **The following command creates a CSR on the `enigma` system:**

```
# ikecert certlocal -kc -m 2048 -t rsa-sha384 \
> -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
...
8qLqDjaStLGfhD00
-----END CERTIFICATE REQUEST-----
```

2. Submit the CSR to a CA.

The CA can tell you how to submit the CSR. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your CSR into the form. When your request has been checked by the organization, the organization issues you signed certificates. For more information, see [“Using Public Key Certificates in IKE” on page 128](#).

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 128](#).

a. Become an administrator.

For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#). If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

b. Add the public key and its certificate that you received from the CA.

```
# ikecert certdb -a < /tmp/PKIcert.eml
```

c. Add the CA's public certificate.

You might also need to add intermediate certificates.

```
# ikecert certdb -a < /tmp/PKIca.eml
```

d. If the CA has sent a list of revoked certificates, add the CRL to the `certldb` database:

```
# ikecert certldb -a
Press the Return key
Paste the CRL
-----BEGIN CRL-----
...
-----END CRL-----
Press the Return key
Press Control-D
```

4. Use the `cert_root` keyword in the `/etc/inet/ike/config` file to identify the CA that issued the certificate.

Use the Distinguished Name (DN) of the CA's certificate.

a. For example, the `ike/config` file on the `partym` system might appear similar to the following:

```
# Trusted root cert
# This certificate is from Example CA
# This is the X.509 distinguished name for the CA's cert

cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example, CN=Example CA"

## Parameters that may also show up in rules.

p1_xform
{ auth_method rsa_sig oakley_group 1 auth_alg sha384 encr_alg aes}
p2_pfs 2

{
  label "US-partym to JA-enigma - Example CA"
  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 must 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=ExampleCA\, Inc., OU=CA-Example, CN=Example CA"
...
{
  label "JA-enigma to US-partym - Example CA"
  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. Set the IKEv1 policies for handling revoked certificates.

Choose the appropriate option:

■ No OCSP available

If the public key certificate provides a URI to reach the OCSP server but your system cannot connect to the Internet, add the keyword `ignore_ocsp` to the `ike/config` file.

```
# Trusted root cert
...
cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example,..."
ignore_ocsp
...
```

The `ignore_ocsp` keyword tells IKEv1 to assume that the certificate is valid.

■ No CRL available

If the CA does not provide a reliable source for CRLs or your system cannot connect to the Internet to retrieve CRLs, add the keyword `ignore_crls` to the `ike/config` file.

```
# Trusted root cert
...
cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example,..."
ignore_crls
...
```

■ URI for CRLs or OCSP available

If the CA provides a central distribution point for revoked certificates, you can modify the `ike/config` file to use the URI.

See [“How to Handle Revoked Certificates in IKEv1” on page 181](#) for examples.

Example 10-2 Using `rsa_encrypt` When Configuring IKEv1

When you use `auth_method rsa_encrypt` in the `ike/config` file, you must add the peer's certificate to the `publickeys` database.

1. Send the certificate to the remote system's administrator.

You can paste the certificate into an email message.

For example, the `partym` administrator would send the following message:

```
To: admin@enigma.ja.example.com
```



```

From: admin@party.us.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MII...
-----END X509 CERTIFICATE-----

```

The enigma administrator would send the following message:

```

To: admin@party.us.example.com
From: admin@enigma.ja.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MII
...
-----END X509 CERTIFICATE-----

```

2. On each system, add the emailed certificate to the local `publickeys` database.

```
# ikercert certdb -a < /tmp/saved.cert.eml
```

The authentication method for RSA encryption hides identities in IKE from eavesdroppers. Because the `rsa_encrypt` method hides the peer's identity, IKEv1 cannot retrieve the peer's certificate. As a result, the `rsa_encrypt` method requires that the IKEv1 peers know each other's public keys.

Therefore, when you use an `auth_method` of `rsa_encrypt` in the `/etc/inet/ike/config` file, you must add the peer's certificate to the `publickeys` database. The `publickeys` database then holds at least three certificates for each communicating pair of systems:

- Your public key certificate
- The CA's certificate chain
- The peer's public key certificate

Troubleshooting – The IKEv1 payload, which includes at least three certificates, can become too large for `rsa_encrypt` to encrypt. Errors such as “authorization failed” and “malformed payload” can indicate that the `rsa_encrypt` method cannot encrypt the total payload. Reduce the size of the payload by using a method, such as `rsa_sig`, that requires only two certificates.

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Generate and Store Public Key Certificates for IKEv1 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 IKEv1 to Find the Sun Crypto Accelerator 6000 Board” on page 191](#) for setup instructions.

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#).

If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

1. Generate a self-signed certificate or a CSR, and specify the token ID.

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.

Choose one of the following options:

- **For a self-signed certificate, use this syntax:**

```
# ikecert certlocal -ks -m 2048 -t rsa-sha512 \  
> -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 CSR, use this syntax:**

```
# ikecert certlocal -kc -m 2048 -t rsa-sha512 \
> -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 username, 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 - If you type the `ikecert` commands with the `-p` option, the PKCS #11 token is stored on disk *as clear text* and is protected by root permissions. If you do not store the PIN on disk, you must unlock the token by using the `ikedm` command after the `in.iked` command is running.

After you type the password, the certificate prints the following output:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
oKUDBbZ90/pLWYGr
-----END X509 CERTIFICATE-----
```

3. Send your certificate to 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 message.

- **Send the CSR to a [certificate authority \(CA\)](#).**

Follow the instructions of the CA to submit the certificate request. For a more detailed discussion, see [Step 2](#) of “[How to Configure IKEv1 With Certificates Signed by a CA](#)” on page 173.

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-enigma 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 CA 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 CA
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExampleCA\, Inc., OU=CA-Example, CN=Example CA"

...
{
  label "JA-enigma to US-partym - Example CA"
  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 CA.**

Add the certificate that the CA generated from your certificate request and organization's certificate.

You might also need to add intermediate certificates.

```
# 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 CA, see [“How to Handle Revoked Certificates in IKEv1” on page 181](#).

Next Steps If you have not completed establishing IPsec policy, return to the IPsec procedure to enable or refresh IPsec policy. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

▼ How to Handle Revoked Certificates in IKEv1

Revoked certificates are certificates that are compromised for some reason. A revoked certificate that is in use is a security risk. You have options when verifying certificate revocation. You can use a static list or you can verify revocations dynamically over the HTTP protocol. You have four ways to handle revoked certificates.

- You can instruct IKEv1 to ignore CRLs or OCSP whose uniform resource indicator (URI) is embedded in the certificate. This option is shown in [Step 5](#).

- You can instruct IKEv1 to access the CRLs or OCSP from a URI whose address is embedded in the public key certificate from the CA.
- You can instruct IKEv1 to access the CRLs from an LDAP server whose DN (directory name) entry is embedded in the public key certificate from the CA.
- You can provide the CRL as an argument to the `ikecert certfdb` command. For an example, see [Example 10-3](#).

Before You Begin You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”.

1. Display the certificate that you received from the CA.

For information about the arguments to the `ikecert certfdb` command, see the [ikecert\(1M\)](#) man page.

For example, the following certificate was issued by a company's PKI. Details have been altered.

```
# ikecert certfdb -lv cert-protect.example.com
Certificate Slot Name: 0 Type: dsa-sha256
  (Private key in certlocal slot )
Subject Name: <O=Example, CN=cert-protect.example.com>
Issuer Name: <CN=ExampleCo CO (Cl B), O=Example>
SerialNumber: 14000D93
Validity:
  Not Valid Before: 2013 Sep 19th, 21:11:11 GMT
  Not Valid After: 2017 Sep 18th, 21:11:11 GMT
Public Key Info:
  Public Modulus (n) (2048 bits): C575A...A5
  Public Exponent (e) ( 24 bits): 010001
Extensions:
  Subject Alternative Names:
    DNS = cert-protect.example.com
  Key Usage: DigitalSignature KeyEncipherment
  [CRITICAL]
CRL Distribution Points:
  Full Name:
    URI = #Ihttp://www.example.com/pki/pkismica.crl#i
    DN = <CN=ExampleCo CO (Cl B), O=Example>
  CRL Issuer:
  Authority Key ID:
  Key ID: 4F ... 6B
  SubjectKeyID: A5 ... FD
Certificate Policies
  Authority Information Access
```

Notice the CRL Distribution Points entry.

- The URI entry indicates that this organization's CRL is available on the web.
- The DN entry indicates that the CRL is available on an LDAP server. Once accessed by IKE, the CRL is cached for further use.

To access the CRL, you need to reach a distribution point.

2. Choose one of the following methods to access the CRL from a central distribution point.

■ **Use the URI.**

Add the keyword `use_http` to the host's `/etc/inet/ike/config` file. For example, the `ike/config` file would appear similar to the following:

```
# Use CRL or OCSP from organization's URI
use_http
...
```

■ **Use a web proxy.**

Add the keyword `proxy` to the `ike/config` file. The `proxy` keyword takes a URL as an argument, as in the following:

```
# Use web proxy to reach CRLs or OCSP
proxy "http://proxy1:8080"
```

■ **Use an LDAP server.**

Name the LDAP server as an argument to the `ldap-list` keyword in the host's `/etc/inet/ike/config` file. Your organization provides the name of the LDAP server. The entry in the `ike/config` file would appear similar to the following:

```
# Use CRL from organization's LDAP
ldap-list "ldap1.example.com:389,ldap2.example.com"
...
```

IKE retrieves the CRL and caches the CRL until the certificate expires.

Example 10-3 Pasting a CRL Into the Local `cert1db` Database for IKEv1

If the CA's CRL is not available from a central distribution point, you can add the CRL manually to the local `cert1db` database. Follow the CA's instructions for extracting the CRL into a file, then add the CRL to the database with the `ikecert cert1db -a` command.

```
# ikecert cert1db -a < ExampleCo.Cert.CRL
```

Configuring IKEv1 for Mobile Systems

IPsec and IKE require a unique ID to identify source and destination. For off-site or mobile systems that do not have a unique IP address, you must use another ID type. ID types such as DNS, DN, or email can be used to uniquely identify a system.

Off-site or mobile systems that have unique IP addresses are still best configured with a different ID type. For example, if the systems attempt to connect to a central site from behind a NAT box, their unique addresses are not used. A NAT box assigns an arbitrary IP address, which the central system would not recognize.

Preshared keys also do not work well as an authentication mechanism for mobile systems, because preshared keys require fixed IP addresses. Self-signed certificates, or certificates from a CA enable mobile systems to communicate with the central site.

The following task map lists procedures to configure IKEv1 to handle systems that log in remotely to a central site.

TABLE 10-2 Configuring IKEv1 for Mobile Systems Task Map

Task	Description	For Instructions
Communicate with a central site from off-site.	Enables off-site systems to communicate with a central site. The off-site systems might be mobile.	“How to Configure IKEv1 for Off-Site Systems” on page 184
Use a CA’s public certificate and IKEv1 on a central system that accepts traffic from mobile systems.	Configures a gateway system to accept IPsec traffic from a system that does not have a fixed IP address.	Example 10-4
Use a CA’s public certificate and IKEv1 on a system that does not have a fixed IP address.	Configures a mobile system to protect its traffic to a central site, such as company headquarters.	Example 10-5
Use self-signed certificates and IKEv1 on a central system that accepts traffic from mobile systems.	Configures a gateway system with self-signed certificates to accept IPsec traffic from a mobile system.	Example 10-6
Use self-signed certificates and IKEv1 on a system that does not have a fixed IP address.	Configures a mobile system with self-signed certificates to protect its traffic to a central site.	Example 10-7

▼ How to Configure IKEv1 for Off-Site Systems

Before You Begin You must assume the root role. For more information, see [“Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”](#). If you administer remotely, see [Example 7-1](#) and [“How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2”](#) for secure remote login instructions.

- 1. Configure the central system to recognize mobile systems.**

a. Configure the `ipsecinit.conf` file.

The central system needs a policy that allows a wide range of IP addresses. Later, certificates in the IKE policy ensure that the connecting systems are legitimate.

```
# /etc/inet/ipsecinit.conf on central
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}
```

b. Configure the IKEv1 configuration file.

DNS identifies the central system. Certificates are used to authenticate the system.

```
## /etc/inet/ike/ike.config on central
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server "ldap-server1.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, *central*.

```
# /etc/hosts on mobile
central 192.xxx.xxx.x
```

b. Configure the `ipsecinit.conf` file.

The mobile system needs to find the central system by its public IP address. The systems must configure the same IPsec policy.

```
# /etc/inet/ipsecinit.conf on mobile
# Find central
{raddr 192.xxx.xxx.x} ipsec {encr_algs aes encr_auth_algs sha256 sa shared}
```

c. Configure the IKEv1 configuration file.

The identifier cannot be an IP address. The following identifiers are valid for mobile systems:

- `DN=ldap-directory-name`
- `DNS=domain-name-server-address`
- `email=email-address`

Certificates are used to authenticate the mobile system, *mobile*.

```
## /etc/inet/ike/ike.config on mobile
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server "ldap-server1.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:default` service.

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

Example 10-4 Configuring a Central Computer That Uses IKEv1 to Accept Protected Traffic From a Mobile System

IKE can initiate negotiations from behind a NAT box. However, the ideal setup for IKE is without an intervening NAT box. In the following example, the CA's public certificate has been placed on the mobile system and the central system. A central system accepts IPsec negotiations from a system behind a NAT box. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see [Example 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=ExampleCA Inc, OU=CA-Example, CN=Example CA"

```

```

#
# Rule for off-site systems with root certificate
{
  label "Off-site system with root certificate"
  local_id_type DNS
  local_id "main1.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 and IKEv1

In the following example, the CA's public certificate is placed on the mobile system and the central system. mobile1 is connecting to the company headquarters from home. The Internet service provider (ISP) network uses a NAT box to enable the ISP to assign mobile1 a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. For setting up the computer at company headquarters, see [Example 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=ExampleCA Inc, OU=CA-Example, CN=Example CA"
#
# Rule for off-site systems with root certificate
{
    label "Off-site mobile1 with root certificate"
    local_id_type DNS
    local_id "mobile1.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. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

Configuring IKEv1 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 IKEv1 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 Product Library Documentation \(http://docs.oracle.com/cd/E19321-01/index.html\)](http://docs.oracle.com/cd/E19321-01/index.html).

You must become an administrator who is assigned the Network IPsec Management rights profile. For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2”.

If you administer remotely, see [Example 7-1](#) and “How to Remotely Administer ZFS With Secure Shell” in “Managing Secure Shell Access in Oracle Solaris 11.2” for secure remote login instructions.

1. Verify that the PKCS #11 library is linked.

IKEv1 uses the library's routines to handle key generation and key storage on the Sun Crypto Accelerator 6000 board.

```
$ 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 IKEv1 keys.

For instructions on how to use the token, see “How to Generate and Store Public Key Certificates for IKEv1 in Hardware” on page 178.

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. For examples of IPsec policy protecting VPNs, see [“Protecting a VPN With IPsec” on page 108](#). For other examples of IPsec policy, see [“How to Secure Network Traffic Between Two Servers With IPsec” on page 102](#).

Troubleshooting IPsec and Its Key Management Services

This chapter describes how to troubleshoot IPsec and its keys, how to view configuration information, and how to view information about active IPsec, IKE, and manual key services.

The chapter contains the following information:

- [“Troubleshooting IPsec and Its Key Management Configuration” on page 193](#)
- [“Viewing Information About IPsec and Its Keying Services” on page 201](#)
- [“Managing IPsec and Its Keying Services” on page 206](#)
- [“Managing the Running IKE Daemons” on page 208](#)

Troubleshooting IPsec and Its Key Management Configuration

You can set up your system for troubleshooting before or during a problem that needs troubleshooting.

When troubleshooting, you can run many commands in a profile shell as an administrator with the Network IPsec Management rights profile. However, to read logs you must assume the root role.

The prompts in the troubleshooting sections indicate whether you must have rights to run a command.

- # prompt – A user with the appropriate administrative rights or a role with those rights can run the command.
- % prompt – A regular user can run the command.

▼ How to Prepare IPsec and IKE Systems for Troubleshooting

Before you enable IPsec and its key management services, you can set up your system with logs and tools that aid in troubleshooting.

1. Locate the logs for the IPsec and IKEv2 services.

The `-L` option provides the full path to the logs. These logs contain information messages as well as error messages.

```
% svcs -L policy
/var/svc/log/network-ipsec-policy:default.log

% svcs -L ikev2
/var/svc/log/network-ipsec-ike:ikev2.log
```

2. Configure a debug log file for IKEv2.

The root role can read these logs.

```
% svccfg -s ikev2 listprop | grep debug
config/debug_level          astring      op
config/debug_logfile       astring      /var/log/ikev2/in.ikev2d.log
```

The levels of debugging are described in the [ikeadm\(1M\)](#) man page. The values `verbose` and `all` are useful when troubleshooting.

3. (Optional) Configure the debug level.

The following command sets the debug level permanently. To set the debug level temporarily, see [Example 11-3](#).

```
# svccfg -s ikev2 setprop config/debug_level = all
```

If the `ikev2` service is enabled, it must be refreshed to use the new debug level.

```
# svcadm refresh ikev2
```

4. (Optional) Install the wireshark package.

The Wireshark application can read snoop output.

```
% pkg info -r wireshark
Name: diagnostic/wireshark
Summary: Graphical network protocol analyzer
Category: Applications/Internet
State: Not installed
Publisher: solaris
...
FMRI: pkg://solaris/diagnostic/wireshark@version
```

```
# pkg install diagnostic/wireshark
```

▼ How to Troubleshoot Systems Before IPsec and IKE Are Running

You can check the syntax of the IPsec configuration file, the IPsec keys file, and the validity of certificates in the keystore before running the services.

1. Verify the syntax of the IPsec configuration file.

```
# ipsecconf -c /etc/inet/ipsecinit.conf
ipseconf: Invalid pattern on line 5: ukp
ipseconf: form_ipsec_conf error
ipseconf: Malformed command (fatal):
{ ukp 58 type 133-137 dir out} pass {}
```

```
ipseconf: 1 policy rule(s) contained errors.
ipseconf: Fatal error - exiting.
```

If the output shows an error, fix it and run the command until the verification succeeds.

2. Verify the syntax of the ipseckeys file.

```
# ipseckey -c /etc/inet/secret/ipseckeys
Config file /etc/inet/secret/ipseckeys has insecure permissions,
will be rejected in permanent config.
```

If the output shows an error, fix the error then refresh the service.

```
# svcadm refresh ipsec/policy
```

Note - The IKE configuration files and IKE preshared key files are validated by a running IKE daemon.

3. Verify the validity of the certificates.

- To verify the validity of self-signed certificates in IKEv2, perform [Step 4](#) in “[How to Configure IKEv2 With Self-Signed Public Key Certificates](#)” on page 145.
- To verify that a public key certificate is not revoked in IKEv2, follow the procedure “[How to Set a Certificate Validation Policy in IKEv2](#)” on page 153.
- To verify the validity of self-signed certificates in IKEv1, perform [Step 4](#) in “[How to Configure IKEv1 With Self-Signed Public Key Certificates](#)” on page 168.
- To verify that a public key certificate is not revoked in IKEv1, follow the procedure “[How to Handle Revoked Certificates in IKEv1](#)” on page 181.

Next Steps If your configuration does not work when you enable IPsec and its keying services, you must troubleshoot while the services are running.

▼ How to Troubleshoot Systems When IPsec Is Running

On running systems that are exchanging or attempting to exchange packets by using IKE, you can use the `ikeadm` command to view statistics, rules, preshared keys and other things. You can also use the log files and selected tools, such as the Wireshark application.

1. Investigate the following items:

■ Verify that the policy and appropriate key management services are enabled.

On the following test system, the `manual-key` service is being used for key management:

```
% svcs -a | grep ipsec
online      Feb_04   svc:/network/ipsec/manual-key:default
online      Feb_04   svc:/network/ipsec/ipsecalgs:default
online      Feb_04   svc:/network/ipsec/policy:default
disabled    Feb_28   svc:/network/ipsec/ike:ikev2
disabled    Feb_28   svc:/network/ipsec/ike:default
```

If the service is disabled, enable it.

You can use both IKE services concurrently. You can also use manual keys and IKE concurrently, but this configuration could result in oddities that are difficult to troubleshoot.

■ View the end of the log file for the IKEv2 service.

```
# svcs -xL ikev2
svc:/network/ipsec/ike:ikev2 (IKEv2 daemon)
State: disabled since October 10, 2013 10:10:40 PM PDT
Reason: Disabled by an administrator.
See: http://support.oracle.com/msg/SMF-8000-05
See: in.ikev2d(1M)
See: /var/svc/log/network-ipsec-ike:ikev2.log
Impact: This service is not running.
Log:
Oct 01 13:20:20: (1) Property "debug_level" set to: "op"
Oct 01 13:20:20: (1) Errors and debug messages will be written to:
/var/log/ikev2/in.ikev2d.log
[ Oct 10 10:10:10 Method "start" exited with status 0. ]
[ Oct 10 10:10:40 Stopping because service disabled. ]
[ Oct 10 10:10:40 Executing stop method (:kill). ]

Use: 'svcs -Lv svc:/network/ipsec/ike:ikev2' to view the complete log.
```

- **(Optional) You can set a temporary value for the debug level of the running daemon.**

```
# ikeadm set debug verbose /var/log/ikev2/in.ikev2d.log
Successfully changed debug level from 0x80000000 to 0x6204
Debug categories enabled:
    Operational / Errors
    Config file processing
    Interaction with Audit
    Verbose Operational
```

2. **Verify that the output of the ipsecconf command matches the contents of the policy file.**

```
# ipsecconf
#INDEX 14
...
{ laddr 10.133.66.222 raddr 10.133.64.77 }
ipsec { encr_algs aes(256) encr_auth_algs sha512 sa shared }
...
{ laddr 10.134.66.122 raddr 10.132.55.55 }
ipsec { encr_algs aes(256) encr_auth_algs sha512 sa shared }

# cat /etc/inet/ipsecinit.conf
...
{ laddr 10.133.66.222 raddr 10.133.64.77 }
ipsec { encr_algs aes(256) encr_auth_algs sha512 sa shared }

{ laddr 10.134.66.122 raddr 10.132.55.55 }
ipsec { encr_algs aes(256) encr_auth_algs sha512 sa shared }
```

Note - Wildcard addresses can obscure a match, so verify that any specific addresses in the ipsecinit.conf file are within the range of wildcard addresses in the output of ipsecconf.

If no output prints for the ipsecconf command, verify that the policy service is enabled and refresh the service.

```
% svcs policy
STATE      STIME    FMRI
online     Apr_10   svc:/network/ipsec/policy:default
```

If the output shows an error, edit the ipsecinit.conf file to fix the error then refresh the service.

3. **Validate your IKEv2 configuration.**

For configuration output that might require fixing, see [Example 11-1](#) and [Example 11-2](#). The output in the following example indicates that the configuration is valid.

```
# /usr/lib/inet/in.ikev2d -c
```

```
Feb 04 12:08:25: (1) Reading service properties from smf(5) repository.
Feb 04 12:08:25: (1) Property "config_file" set to: "/etc/inet/ike/ikev2.config"
Feb 04 12:08:25: (1) Property "debug_level" set to: "all"
Feb 04 12:08:25: (1) Warning: debug output being written to stdout.
Feb 04 12:08:25: (1) Checking IKE rule #1: "Test 104 to 113"
Feb 04 12:08:25: (1) Configuration file /etc/inet/ike/ikev2.config is valid.
Feb 04 12:08:25: (1) Pre-shared key file /etc/inet/ike/ikev2.preshared is valid.
```

Note - The warning about debug output does not change even after you specify a debug log file. If you specify a value for the `debug_logfile` service property, the warning means that debug output is being delivered to that file. Otherwise, debug output is delivered to the console.

- In the Checking IKE rule lines, verify that the IKE rules connect the appropriate IP addresses. For example, the following entries match. The `laddr` value from the `ipsecinit.conf` file matches the `local_addr` value from the `ikev2.config` file, and the remote addresses match.

```
{ laddr 10.134.64.104 raddr 10.134.66.113 }      /** ipsecinit.conf **/
    ipsec {encr_algs aes encr_auth_algs sha512 sa shared}

local_addr  10.134.64.104                        /** ikev2.config **/
remote_addr 10.134.66.113                        /** ikev2.config **/
```

If the entries do not correspond, fix the configuration to identify the correct IP addresses.

Note - Rules can have wildcard addresses such as `10.134.0.0/16` that cover a range of addresses. Verify the range against specific addresses.

- If the Pre-shared key file line indicates that the file is not valid, fix the file. Check for typographical errors. Also, in IKEv2, check that the label value in the rule in `ikev2.config` matches the label value in the `ikev2.preshared` file. Then, if you are using two keys, verify that the local preshared key on one system matches the remote preshared key on its peer, and that the remote key matches the local key on the peer.

If your configuration still does not work, see [“Troubleshooting IPsec and IKE Semantic Errors” on page 200](#).

Example 11-1 Fixing an Invalid IKEv2 Configuration

In the following output, the lifetime of the IKE SA is too short.

```
# /usr/lib/inet/in.ikev2d -c
...
May 08 08:52:49: (1) WARNING: Problem in rule "Test 104 to 113"
May 08 08:52:49: (1) HARD lifetime too small (60 < 100)
May 08 08:52:49: (1) -> Using 100 seconds (minimum)
May 08 08:52:49: (1) Checking IKE rule #1: "config 10.134.13.113 to 10.134.13.104"
```

...

This value has been explicitly set in the `ikev2.config` file. To remove the warning, change the lifetime value to at least `100` and refresh the service.

```
# pfedit /etc/inet/ike/ikev2.config
...
## childsa_lifetime_secs 60
childsa_lifetime_secs 100
...
# /usr/lib/inet/in.ikev2d -c
...
# svcadm refresh ikev2
```

Example 11-2 Fixing a No Matching Rule Message

In the following output, a preshared key is defined but is not used in a rule.

```
# /usr/lib/inet/in.ikev2d -c
Feb 4 12:58:31: (1) Reading service properties from smf(5) repository.
Feb 4 12:58:31: (1) Property "config_file" set to: "/etc/inet/ike/ikev2.config"
Feb 4 12:58:31: (1) Property "debug_level" set to: "op"
Feb 4 12:58:31: (1) Warning: debug output being written to stdout.
Feb 4 12:58:31: (1) Checking IKE rule #1: "Test 104 to 113"
Feb 4 12:58:31: (1) Configuration file /etc/inet/ike/ikev2.config is valid.
Feb 4 12:58:31: (1) No matching IKEv2 rule for pre-shared key ending on line 12
Feb 4 12:58:31: (1) Pre-shared key file /etc/inet/ike/ikev2.preshared is valid.
```

The output indicates that only one rule exists.

- If the rule requires a preshared key, then the label of the preshared key does not match the label of the rule. Fix the `ikev2.config` rule label and the `ikev2.preshared` key label to match.
- If the rule uses a certificate, then you can remove or comment out the preshared key that ends on line 12 in the `ikev2.preshared` file to prevent the No matching message.

Example 11-3 Setting a New Debug Level on a Running IKE Daemon

In the following output, debug output is set to `all` in the `ikev2` service.

```
# /usr/lib/inet/in.ikev2d -c
Feb 4 12:58:31: (1) Reading service properties from smf(5) repository.
...
Feb 4 12:58:31: (1) Property "debug_level" set to: "all"
...
```

If you have completed [Step 2](#) in [“How to Troubleshoot Systems Before IPsec and IKE Are Running” on page 195](#) and the debug output is still `op` rather than `all`, use the `ikeadm` command to set the debug level on the running IKE daemon.

```
# ikedadm set debug_level all
```

Troubleshooting IPsec and IKE Semantic Errors

If the investigations in [“How to Troubleshoot Systems When IPsec Is Running” on page 196](#) fail to handle the problem, then the semantics of your configuration is the likely problem, rather than the syntax of your files or the service configuration.

- If both the `ike:default` and `ike:ikev2` service instances are enabled, ensure that the IKEv2 and IKEv1 rules do not overlap. Rules that apply to the same network endpoints can result in redundant IPsec SAs and could cause a lack of connectivity in certain situations.

If you change an IKE rule, read the rule into the kernel.

```
# ikeadm -v[1|2] read rule
```

- If you are running IKEv1, make sure that the algorithm mechanisms in your rules are available on the IKEv1 system that you are connecting to. To view the available algorithms, run the `ikeadm dump algorithms` command on the system that does not support IKEv2:

```
# ikeadm dump groups      Available Diffie-Hellman groups
# ikeadm dump encralgs    All IKE encryption algorithms
# ikeadm dump authalgs    All IKE authentication algorithms
```

Correct both the IPsec and IKEv1 policy files to use algorithms that are available on both systems. Then, restart the IKEv1 service and refresh the IPsec service.

```
# svcadm restart ike:default; svcadm refresh ipsec/policy
```

- If you are using preshared keys with IKEv1, and the remote IKEv1 system is rebooted, run the `ipseckey flush` command on the local system.
- If you are using self-signed certificates, verify with the other administrator that a certificate with the same DN has not been re-created and that the hash values of your certificates match. For the verification steps, see [Step 4](#) in [“How to Configure IKEv2 With Self-Signed Public Key Certificates” on page 145](#).

If the certificate is updated, import the new certificate, then refresh and restart the IKEv2 service.

- Use the `ikeadm -v2 dump | get` command to view the current IKEv2 configuration. For a usage summary, see [“Viewing IKE Information” on page 202](#).
- Use the `kstat` command to display IPsec-related statistics. For more information, see the [kstat\(1M\)](#) man page.

```
# kstat -m ipsecesp
# kstat -m ipsecah
# kstat -m ip
```

The `kstat` output in the following example indicates no problems in the `ipsecesp` module.


```
# kstat -m ipsecesp
module: ipsecesp           instance: 0
name:  esp_stat           class:  net
      acquire_requests    18
      bad_auth            0
      bad_decrypt         0
      bad_padding         0
      bytes_expired       0
      crtime              4.87974774
      crypto_async        0
      crypto_failures     0
      crypto_sync         172
      good_auth           86
      keysock_in          135
      num_aalgs           9
      num_ealgs           13
      out_discards        0
      out_requests        86
      replay_early_failures 0
      replay_failures     0
      sa_port_renumbers   0
      snaptime            5946769.7947628
```

- Use the snoop command to view the traffic that is not being protected. The Wireshark application can read snoop output. For an example of snoop output, see [“How to Verify That Packets Are Protected With IPsec”](#) on page 123.

Viewing Information About IPsec and Its Keying Services

Note - For most commands, you must become an administrator who is assigned the Network IPsec Management rights profile. You must be typing in a profile shell. For more information, see [“Using Your Assigned Administrative Rights”](#) in [“Securing Users and Processes in Oracle Solaris 11.2”](#).

Viewing IPsec and Manual Key Service Properties

You can view the name of the IPsec policy file and the file that holds manual keys.

- To show the name of the IPsec configuration file:

```
% svccfg -s policy listprop config/config_file
config/config_file      astring      /etc/inet/ipsecinit.conf
```

- To show the name of the file that holds manual keys for IPsec:

```
% svccfg -s manual-key listprop config/config_file
config/config_file      astring      /etc/inet/secret/ipseckeys
```

Viewing IKE Information

You can view the properties of the IKE service, aspects of the IKE state and IKE daemon object, and certificate validation policy. If you are running both IKE services, you can display information per service or for both services. These commands can be helpful during testing, troubleshooting, and monitoring.

- Viewing the properties of the IKE service instances – The output displays the configurable properties of the IKEv2 service, including the names of the configuration files.

Note - Review the [ipseccfg\(1M\)](#), [in.ikev2d\(1M\)](#), and [in.iked\(1M\)](#) man pages to ensure that you can or should modify a property in the config group of the IPsec, IKEv2, or IKEv1 service. For example, IKEv2 configuration files are created with special permissions and owned by `ikeuser`. The permissions and file owner must not be changed.

```
% svccfg -s ipsec/ike:ikev2 listprop config
config                application
config/allow_keydump  boolean          false
config/config_file    astring          /etc/inet/ike/ikev2.config
config/ignore_errors  boolean          false
config/kmf_policy     astring          /etc/inet/ike/kmf-policy.xml
config/max_child_sas  integer          0
config/max_threads    integer          0
config/min_threads    integer          0
config/preshared_file astring          /etc/inet/ike/ikev2.preshared
config/response_wait_time integer          30
config/value_authorization astring          solaris.smf.value.ipsec
config/debug_logfile  astring
config/debug_level    astring          op
```

The output in the following example displays the configurable properties of the IKEv1 service. Do not specify the `:default` service instance.

```
% svccfg -s ipsec/ike listprop config
config                application
```

```

config/admin_privilege    astring    base
config/config_file       astring    /etc/inet/ike/config
config/debug_level       astring    op
config/debug_logfile     astring    /var/log/in.iked.log
config/ignore_errors     boolean    false
config/value_authorization astring    solaris.smf.value.ipsec

```

- Viewing the current state of the IKE daemon – The output in the following example displays the arguments to the `ikeadm` command. These arguments display the current state of the daemon.

Note - To use the `ikeadm` command, the IKE daemon must be running.

```

% ikeadm help
...
    get  debug|priv|stats|p1|ikesa|rule|preshared|defaults [identifier]
    dump p1|ikesa|rule|preshared|certcache|groups|encr|algs|authalgs
    read rule|preshared [filename]
    help [get|set|add|del|dump|flush|read|write|token|help]

```

- Showing the syntax of a specific argument to the `ikeadm` command – Use the `help` subcommands to show command argument syntax. For example:

```

% ikeadm help read
This command reads a new configuration file into
in.iked, discarding the old configuration info.

```

```

Sets of data that may be read include:
    rule          all phase 1/ikesa rules
    preshared     all preshared keys

```

```

A filename may be provided to specify a source file
other than the default.

```

- Viewing preshared keys – You can view preshared keys for IKEv1 and IKEv2.

Note - If you are running only one IKE version, you can omit the `-v` option.

For IKEv2:

```
# ikeadm -v2 dump preshared
```

For IKEv1:

```
# ikeadm set priv keymat
# ikeadm -v1 dump preshared
```

```
PSKEY: Rule label: "Test PSK 197 to 56"  
PSKEY: Local pre-shared key (80 bytes): 74206272696c6c696720...3/584  
PSKEY: Remote pre-shared key (80 bytes): 74206272696c6c696720...3/584
```

Completed dump of preshared keys

- Viewing IKE SAs – The output includes information about the SA, the transform, the local and remote systems, and other details. If communication has not been requested, no SAs exist, so no information exists to display.

```
# ikeadm -v2 dump ikesa  
IKESA: SPIs: Local 0xd3db95689459cca4 Remote 0xb5878717f5cfa877  
...  
XFORM: Encryption alg: aes-cbc(256..256); Authentication alg: hmac-sha512  
...  
LOCIP: AF_INET: port 500, 10.1.2.3 (example-3).  
...  
REMIP: AF_INET: port 500, 10.1.4.5 (ex-2).  
...  
LIFTM: SA expires in 11459 seconds (3.18 hours)  
...  
STATS: 0 IKE SA rekeys since initial AUTH.  
LOCID: Initiator identity, type FQDN  
...  
CHILD: ESP Inbound SPI: 0x94841ca3, Outbound SPI 0x074ae1e5  
...  
Completed dump of IKE SA info
```

- Viewing active IKE rules – A listed IKE rule might not be in use, but it is available for use.

```
# ikeadm -v2 dump rule  
  
GLOBL: Label 'Test Rule1 for PSK', key manager cookie 1  
GLOBL: Local auth method=pre-shared key  
GLOBL: Remote auth method=pre-shared key  
  
GLOBL: childsa_pfs=false  
GLOBL: authentication_lifetime=86400 seconds (1.00 day)  
GLOBL: childsa_lifetime=120 seconds (2.00 minutes)  
GLOBL: childsa_softlife=108 seconds (1.80 minute)  
GLOBL: childsa_idletime=60 seconds  
GLOBL: childsa_lifetime_kb=122880 kilobytes (120.00 MB)  
GLOBL: childsa_softlife_kb=110592 kilobytes (108.00 MB)  
LOCIP: IP address range(s):  
LOCIP: 10.142.245.197  
REMIP: IP address range(s):  
REMIP: 10.134.64.56
```

```

LOCID: Identity descriptors:
LOCID: Includes:
LOCID:      fqdn="gloria@ms.mag"
REMID: Identity descriptors:
REMID: Includes:
REMID:      fqdn="gloria@ms.mag"
XFRMS: Available Transforms:

XF 0: Encryption alg: aes-cbc(128..256); Authentication alg: hmac-sha512
XF 0: PRF: hmac-sha512 ; Diffie-Hellman Group: 2048-bit MODP (group 14)
XF 0: IKE SA lifetime before rekey: 14400 seconds (4.00 hours)

```

Completed dump of policy rules

- Viewing certificate validation policy in IKEv2 – You must specify the `dbfile` value and the `policy` value.
 - Dynamically downloaded CRLs might require administrator intervention to adjust the responder timeout.

In the output in the following example, the CRLs are downloaded from the URI that is embedded in the certificate, then the lists are cached. When the cache contains an expired CRL, a new CRL is downloaded to replace the old one.

```
# kmfcfg list dbfile=/etc/inet/ike/kmf-policy.xml policy=default
```

...

Validation Policy Information:

```

Maximum Certificate Revocation Responder Timeout: 10
Ignore Certificate Revocation Responder Timeout: true

```

...

CRL:

```

Base filename: [not set]
Directory: /var/user/ikeuser/crls
Download and cache CRL: true
CRL specific proxy override: www-proxy.cagate.example.com:80
Ignore CRL signature: false
Ignore CRL validity date: false

```

```
IPsec policy bypass on outgoing connections: true
```

...

- Statically downloaded CRLs require frequent administrator attention.

When the administrator sets the CRL entries to the following values, the administrator is responsible for manually downloading the CRLs, populating the directory, and maintaining current CRLs:

...

```

Directory: /var/user/ikeuser/crls
Download and cache CRL: false
Proxy: [not set]

```

...

Managing IPsec and Its Keying Services

IPsec policy is enabled by default, but it lacks configuration information.

Key management is not enabled by default. You can configure IKE or manual key management, or both. Each IKE rule indicates which key management service is used. The `ikeadm` command can modify the running IKE daemon.

Configuring and Managing IPsec and Its Keying Services

- Configuring and refreshing IPsec, then viewing policy:

```
# pfedit /etc/inet/ipsecinit.conf
# ipsecconf -c /etc/inet/ipsecinit.conf
# svcadm refresh ipsec/policy
# ipsecconf -Ln
```

- Configuring and enabling manual keys for IPsec:

```
# pfedit -s /etc/inet/secret/ipseckeys
# svcadm enable ipsec/manual-key
```

- Configuring and enabling IKEv2:

```
# pfedit /etc/inet/ike/ikev2.config
# /usr/lib/inet/in.ikev2d -c
# svcadm enable ipsec/ike:ikev2
```

- Configuring and enabling IKEv1:

```
# pfedit /etc/inet/ike/config
# /usr/lib/inet/in.iked -c
# svcadm enable ipsec/ike:default
```

- Verifying that IPsec and IKE are configured on a system where the services are enabled:

```
# ipsecconf -Ln
# ikeadm -v2 dump rule
# ikeadm set priv keymat
# ikeadm -v1 dump rule
```

- Modifying key management:

For IKEv2:

```
# pfedit /etc/inet/ike/ikev2.config
# /usr/lib/inet/in.ikev2d -c
# svcadm restart ipsec/ike:ikev2
```

For IKEv1:

```
# pfedit /etc/inet/ike/config
# /usr/lib/inet/in.iked -c
# svcadm restart ipsec/ike:default
```

For manual key management:

```
# pfedit -s /etc/inet/secret/ipseckeys
# ipseckey -c /etc/inet/secret/ipseckeys
# svcadm refresh ipsec/manual-key
```

- Modifying IPsec and IKE configurable properties:

IPsec service:

```
# svccfg -s ipsec/policy setprop config/property = value
# svcadm refresh ipsec/policy; svcadm restart ipsec/policy
```

IKEv2 service:

```
# svccfg -s ike:ikev2 editprop
# svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
```

IKEv1 service:

```
# svccfg -s ipsec/ike setprop config/property = value
# svcadm refresh ipsec/ike:ikev2; svcadm restart ipsec/ike:ikev2
```

Manual keys service:

```
# svccfg -s ipsec/manual-key setprop config/property = value
# svcadm refresh ipsec/manual-key; svcadm restart ipsec/manual-key
```

- Configuring preshared keys for IKEv2:

```
# pfedit -s /etc/inet/ike/ikev2.preshared
# /usr/lib/inet/in.ikev2d -c
# svcadm restart ikev2
```

- Configuring preshared keys for IKEv1:

```
# pfedit -s /etc/inet/secret/ike.preshared
# svcadm restart ike
```

Managing the Running IKE Daemons

For more information, review the [ikeadm\(1M\)](#) man page. The commands in this section are available only when the IKEv2 or IKEv1 daemon is running.

- Modifying the running IKE daemon:

The following output displays the arguments to the `ikeadm` command that can modify the current state of the daemon. Some arguments are specific to the IKEv2 or the IKEv1 daemon.

```
% ikeadm help
...
    set  priv level
    set  debug level [filename]
    add  rule|preshared {definition}|filename
    del  pl|ikesa|rule|preshared identifier
    flush pl|ikesa|certcache
    write rule|preshared filename
    token login|logout PKCS#11-Token-Object
```

- Showing the syntax of a specific argument to the `ikeadm` command:

```
% ikeadm help add
This command adds items to in.iked's tables.

Objects that may be set include:
    rule          a phase 1 or IKE SA policy rule
    preshared     a preshared key
```

Objects may be entered on the command-line, as a series of keywords and tokens contained in curly braces ('{', '}'); or the name of a file containing the object definition may be provided.

For security purposes, preshared keys may only be entered on the command-line if `ikeadm` is running in interactive mode.

- Modifying the IKEv2 daemon with the `ikeadm` command:

```
# ikeadm add rule | preshared {definition} | filename
# ikeadm flush ikesa
# ikeadm del ikesa | rule | preshared identifier
# ikeadm set debug level
# ikeadm token login | logout PKCS#11-Token-Object
```



```
# ikeadm write rule | preshared filename
```

- Modifying the IKEv1 daemon with the `ikeadm` command:

```
# ikeadm set debug level
```

```
# ikeadm set privlevel
```

```
# ikeadm add rule | preshared {definition} | filename
```

```
# ikeadm del p1 | rule | preshared identifier
```

```
# ikeadm flush p1 | certcache
```

```
# ikeadm del rule | preshared id
```

```
# ikeadm write rule | preshared filename
```


IPsec and Key Management Reference

This chapter contains reference information about IPsec, IKEv2, and IKEv1.

- [“IPsec Reference” on page 211](#)
- [“IKEv2 Reference” on page 216](#)
- [“IKEv1 Reference” on page 220](#)

For instructions on how to implement IPsec on your network, see [Chapter 7, “Configuring IPsec”](#). For an overview of IPsec, see [Chapter 6, “About IP Security Architecture”](#).

For instructions on implementing IKE, see [Chapter 9, “Configuring IKEv2”](#). For overview information, see [Chapter 8, “About Internet Key Exchange”](#).

IPsec Reference

IPsec Services, Files, and Commands

This section lists the IPsec services, selected IPsec RFCs, and the files and commands that are relevant to IPsec.

IPsec Services

The Service Management Facility (SMF) provides the following services for IPsec:

- `svc:/network/ipsec/policy` service – Manages IPsec policy. By default, this service is enabled. The value of the `config_file` property determines the location of the `ipsecinit.conf` file. The initial value on a system that is running the DefaultFixed network configuration profile is `/etc/inet/ipsecinit.conf`. On systems that are not running this profile, the property value is empty.
- `svc:/network/ipsec/ipsecalgs` service – Manages the algorithms that are available to IPsec. By default, this service is enabled.

- `svc:/network/ipsec/manual-key` service – Activates manual key management. By default, this service is disabled. The value of the `config_file` property determines the location of the `ipseckeys` configuration file. The initial value is `/etc/inet/secret/ipseckeys`.
- `svc:/network/ipsec/ike` service – Manages IKE. By default, this service is disabled. For the configurable properties, see [“IKEv2 Service” on page 217](#) and [“IKEv1 Service” on page 221](#).

For information about SMF, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in [“Managing System Services in Oracle Solaris 11.2”](#). Also see the [`smf\(5\)`](#), [`svcadm\(1M\)`](#), and [`svccfg\(1M\)`](#) man pages.

ipseccnf Command

You use the `ipseccnf` command to configure the IPsec policy for a host. When you run the command to configure the policy, the system creates the IPsec policy entries in the kernel. The system uses these entries to check the policy on all inbound and outbound IP packets. Packets that are not tunneled and forwarded are not subjected to policy checks that are added by using this command. The `ipseccnf` command also manages the IPsec entries in the [security policy database \(SPD\)](#). For IPsec policy options, see the [`ipseccnf\(1M\)`](#) man page.

You must assume the root role to invoke the `ipseccnf` command. The command can configure entries that protect traffic in both directions. The command also can configure entries that protect traffic in only one direction.

Policy entries with a format of local address and remote address can protect traffic in both directions with a single policy entry. For example, entries that contain the patterns `laddr host1` and `raddr host2` protect traffic in both directions if no direction is specified for the named host. Thus, you need only one policy entry for each host.

Policy entries that are added by the `ipseccnf` command are not persistent over a system reboot. To ensure that the IPsec policy is active when the system boots, add the policy entries to the `/etc/inet/ipsecinit.conf` file, then refresh or enable the `policy` service. For examples, see [“Protecting Network Traffic With IPsec” on page 101](#).

ipsecinit.conf Configuration File

To enable the IPsec security policy when you start Oracle Solaris, you create a configuration file to initialize IPsec with your specific IPsec policy entries. The default name for this file is /

etc/inet/ipsecinit.conf. See the [ipseconf\(1M\)](#) 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 ipseconf(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 do not match the above rules to pass up the IP
# network stack. , , ,
```

Security Considerations for ipsecinit.conf and ipseconf

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. For more information, see the SECURITY section of the [ipseccomp\(1M\)](#) man page.

ipsecalgs Command

The Cryptographic Framework provides authentication and encryption algorithms to IPsec. The `ipsecalgs` command can list the algorithms that each IPsec protocol supports. The `ipsecalgs` configuration is stored in the `/etc/inet/ipsecalgs` file. Typically, this file does not need to be modified and must never be edited directly. However, if you need to modify the file, use the `ipsecalgs` command. The supported algorithms are synchronized with the kernel at system boot by the `svc:/network/ipsec/ipsecalgs:default` service.

The valid IPsec protocols and algorithms are described by the ISAKMP [domain of interpretation \(DOI\)](#), which is covered by RFC 2407. Specifically, the ISAKMP DOI defines the naming and numbering conventions for the valid IPsec algorithms and for their protocols, `PROTO_IPSEC_AH` and `PROTO_IPSEC_ESP`. Each algorithm is associated with exactly one protocol. These ISAKMP DOI definitions are in the `/etc/inet/ipsecalgs` file. The algorithm and protocol numbers are defined by the Internet Assigned Numbers Authority (IANA). The `ipsecalgs` command makes the list of algorithms for IPsec extensible.

For more information about the algorithms, refer to the [ipsecalgs\(1M\)](#) man page. For more information about the Cryptographic Framework, see [Chapter 1, “Cryptographic Framework,”](#) in [“Managing Encryption and Certificates in Oracle Solaris 11.2”](#).

ipseckey Command

The `ipseckey` command with various options manages keys for IPsec manually. For a description of the `ipseckey` command, see the [ipseckey\(1M\)](#) man page.

Security Considerations for ipseckey

The `ipseckey` command enables a role with the Network Security or Network IPsec Management rights profile to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic.

Note - Use IKE rather than manual keying, if possible.

For more information, see the SECURITY section of the [ipseckey\(1M\)](#) man page.

kstat Command

The `kstat` command can display statistics about ESP, AH, and other IPsec data. The IPsec-related options are listed in “[Troubleshooting IPsec and IKE Semantic Errors](#)” on page 200. See also the [kstat\(1M\)](#) man page.

snoop Command and IPsec

The `snoop` command can parse AH and ESP headers. Because ESP encrypts its data, the `snoop` command cannot see encrypted headers that are protected by ESP. AH does not encrypt data, so traffic that is protected by AH can be inspected with the `snoop` command. The `-v` option to the command shows when AH is in use on a packet. For more details, see the [snoop\(1M\)](#) 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 123.

Third-party network analyzers are also available, such as the free open-source software [Wireshark](http://www.wireshark.org/about.html) (<http://www.wireshark.org/about.html>), which is bundled with this release.

IPsec RFCs

The Internet Engineering Task Force (IETF) has published a number of Requests for Comment (RFCs) that describe the security architecture for the IP layer. For a link to the RFCs, see <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 (IKEv1),” November 1998
- RFC 5996, “Internet Key Exchange Protocol Version 2 (IKEv2),” September 2010

- RFC 3554, “On the Use of Stream Control Transmission Protocol (SCTP) with IPsec,” July 2003

Security Associations Database for IPsec

Information on key material for IPsec security services is maintained in a security associations database ([SADB](#)). Security associations (SAs) protect inbound packets and outbound packets.

The `in.iked` daemon and the `ipseckey` command use the `PF_KEY` socket interface to maintain SADB. For more information on how SADB handles requests and messages, see the [pf_key\(7P\)](#) man page.

Key Management in IPsec

The Internet Key Exchange (IKE) protocol handles key management for IPsec automatically. IPsec SAs can also be managed manually with the `ipseckey` command, but IKE is recommended. For more information, see “[Key Management for IPsec Security Associations](#)” on page 89.

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

- `svc:/network/ipsec/ike` service – The SMF service for automatic key management. The `ike` service has two instances. The `ike:ikev2` service instance runs the `in.ikev2d` daemon (IKEv2) to provide automatic key management. The `ike:default` service runs the `in.iked` daemon (IKEv1). For a description of IKE, see [Chapter 8, “About Internet Key Exchange”](#). For more information about the daemons, see the [in.ikev2d\(1M\)](#) and [in.iked\(1M\)](#) man pages.
- `svc:/network/ipsec/manual-key:default` service – The SMF service for manual key management. The `manual-key` service runs the `ipseckey` command with various options to manage keys manually. For a description of the `ipseckey` command, see the [ipseckey\(1M\)](#) man page.

IKEv2 Reference

IKEv2 supersedes IKEv1. For a comparison, see “[Comparison of IKEv2 and IKEv1](#)” on page 130.

IKEv2 Utilities and Files

The following table summarizes the configuration files for IKEv2 policy, the storage locations for IKEv2 keys, and the various commands and services that implement IKEv2. For more about services, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in “[Managing System Services in Oracle Solaris 11.2](#)”.

TABLE 12-1 IKEv2 Service Name, Commands, Configuration and Key Storage Locations, and Hardware Devices

File, Location, Command, or Service	Description	Man Page
<code>svc:/network/ipsec/ike:ikev2</code>	The SMF service that manages IKEv2.	smf(5)
<code>/usr/lib/inet/in.ikev2d</code>	Internet Key Exchange (IKE) daemon. Activates automated key management when the <code>ike:ikev2</code> service is enabled.	in.ikev2d(1M)
<code>/usr/sbin/ikeadm [-v 2]</code>	IKE administration command for viewing and temporarily modifying the IKEv2 policy. Enables you to view IKEv2 administrative objects, such as available Diffie-Hellman groups.	ikeadm(1M)
<code>/usr/sbin/ikev2cert</code>	Certificate database management command for creating and storing public key certificates as the configuration owner, <code>ikeuser</code> . Calls the <code>pktool</code> command.	ikev2cert(1M) pktool(1)
<code>/etc/inet/ike/ikev2.config</code>	Default configuration file for the IKEv2 policy. Contains the site's rules for matching inbound IKEv2 requests and preparing outbound IKEv2 requests. If this file exists, the <code>in.ikev2d</code> daemon starts when the <code>ike:ikev2</code> service is enabled. You can change the location of this file by using the <code>svccfg</code> command.	ikev2.config(4)
<code>/etc/inet/ike/ikev2.preshared</code>	Contains secret keys that two IKEv2 instances that are not using certificate-based authentication can use to authenticate each other.	ikev2.preshared(4)
<code>softtoken keystore</code>	Contains the private keys and public key certificates for IKEv2, owned by <code>ikeuser</code> .	pkcs11_softtoken(5)

IKEv2 Service

The Service Management Facility (SMF) provides the `svc:/network/ipsec/ike:ikev2` service instance to manage IKEv2. By default, this service is disabled. Before enabling this service, you must create a valid IKEv2 configuration in the `/etc/inet/ike/ikev2.config` file.

The following `ike:ikev2` service properties are configurable:

- **config_file property** – Specifies the location of the IKEv2 configuration file. The initial value is `/etc/inet/ike/ikev2.config`. This file has special permissions and must be owned by `ikeuser`. Do not use a different file.

- **debug_level property** – Sets the debugging level of the `in.ikev2d` daemon. The initial value is `op`, or operational. For possible values, see the table on debug levels under Object Types in the [ikeadm\(1M\)](#) man page.
- **debug_logfile property** – Specifies the location of the log file for debugging IKEv2. The initial value is `/var/log/ikev2/in.ikev2d.log`.
- **kmf_policy property** – Sets the location of the log file for certificate policy. The default value is `/etc/inet/ike/kmf-policy.xml`. This file has special permissions and must be owned by `ikeuser`. Do not use a different file.
- **pkcs11_token/pin property** – Sets the PIN to use to log in to the keystore when the IKEv2 daemon starts. This value must match the value that you set for the token with the `ikev2cert setpin` command.
- **pkcs11_token/uri property** – Sets the PKCS #11 URI to the keystore. To use the hardware storage on a crypto accelerator card, you must provide this value.

For information about SMF, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in [“Managing System Services in Oracle Solaris 11.2”](#). Also see the [smf\(5\)](#), [svcadm\(1M\)](#), and [svccfg\(1M\)](#) man pages.

IKEv2 Daemon

The `in.ikev2d` daemon automates the management of cryptographic keys for IPsec on an Oracle Solaris system. The daemon negotiates with a remote system that is running the same protocol to provide authenticated keying materials for security associations (SAs) in a protected manner. The daemon must be running on all systems that plan to use IPsec to protect communications by using the IKEv2 protocol.

By default, the `svc:/network/ipsec/ike:ikev2` service is not enabled. After you have configured the `/etc/inet/ike/ikev2.config` file and enabled the `ike:ikev2` service instance, SMF starts the `in.ikev2d` daemon at system boot.

When the IKEv2 daemon runs, the system authenticates itself to its peer IKEv2 entity and establishes the session keys. At an interval specified in the configuration file, the IKE keys are replaced automatically. The `in.ikev2d` daemon listens for incoming IKE requests from the network and for requests for outbound traffic through the `PF_KEY` socket. For more information, see the [pf_key\(7P\)](#) man page.

Two commands support the IKEv2 daemon. The `ikeadm` command can be used to view the IKE policy. For more information, see [“ikeadm Command for IKEv2” on page 219](#). The `ikev2cert` command enables you to view and manage public and private key certificates. For more information, see [“IKEv2 ikev2cert Command” on page 220](#).

IKEv2 Configuration File

The IKEv2 configuration file, `/etc/inet/ike/ikev2.config`, manages the rules that are used to negotiate the keys for the specified network endpoints that are being protected in the IPsec policy file, `/etc/inet/ipsecinit.conf`.

Key management with IKE includes rules and global parameters. An IKE rule identifies the systems or networks that the keying material secures. The rule also specifies the authentication method. Global parameters include such items as the default amount of time before an IKEv2 SA is rekeyed, `ikesa_lifetime_secs`. For examples of IKEv2 configuration files, see [“Configuring IKEv2 With Preshared Keys” on page 136](#). For examples and descriptions of IKEv2 policy entries, see the `ikev2.config(4)` man page.

The IPsec SAs that IKEv2 supports protect the IP packets according to the policies in the IPsec configuration file, `/etc/inet/ipsecinit.conf`.

The security considerations for the `ike/ikev2.config` file are similar to the considerations for the `ipsecinit.conf` file. For details, see [“Security Considerations for ipsecinit.conf and ipsecconf” on page 213](#).

ikeadm Command for IKEv2

When the `in.ikev2d` daemon is running, you can use the `ikeadm [-v2]` command to do the following:

- View aspects of the IKEv2 state.
- Display IKEv2 daemon objects, such as policy rules, preshared keys, available Diffie-Hellman groups, encryption and authentication algorithms, and existing active IKEv2 SAs.

For examples and a full description of this command's options, see the `ikeadm(1M)` man page.

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. For details, see [“Security Considerations for ipseckey” on page 214](#).

IKEv2 Preshared Keys File

The `/etc/inet/ike/ikev2.preshared` file contains the preshared keys that are used by the IKEv2 service. The file is owned by `ikeuser` and protected at `0600`.

You must customize the default `ikev2.preshared` file when you configure a rule in the `ike/ikev2.config` file that requires preshared keys. Because IKEv2 uses these preshared keys to

authenticate IKEv2 peers, this file must be valid before the `in.ikev2d` daemon reads any rules that require preshared keys.

IKEv2 `ikev2cert` Command

The `ikev2cert` command is used to generate, store, and manage public and private keys and certificates. You use this command when the `ike/ikev2.config` file requires public key certificates. Because IKEv2 uses these certificates to authenticate IKEv2 peers, the certificates must be in place before the `in.ikev2d` daemon reads rules that require the certificates.

The `ikev2cert` command calls the `pktool` command as `ikeuser`.

The following `ikev2cert` commands manage certificates for IKEv2. The commands must be run by the `ikeuser` account. The results are stored in the PKCS #11 softtoken keystore.

- `ikev2cert setpin` – Generates a PIN for the `ikeuser` user. This PIN is required when you use certificates.
- `ikev2cert gencert` – Generates a self-signed certificate.
- `ikev2cert gencsr` – Generates a certificate signing request (CSR).
- `ikev2cert list` – Lists certificates in the keystore.
- `ikev2cert export` – Exports certificates to a file for export.
- `ikev2cert import` – Imports a certificate or CRL.

For information about the syntax of the `ikev2cert` subcommands, see the [pktool\(1\)](#) man page. For examples, see the [ikev2cert\(1M\)](#) man page. For information about the softtoken keystore, see the [cryptoadm\(1M\)](#) man page.

IKEv1 Reference

The following sections provide reference information about IKEv1. IKEv1 is superseded by IKEv2, which offers faster automated key management. For more information about IKEv2, see [“IKEv2 Reference” on page 216](#). For a comparison, see [“Comparison of IKEv2 and IKEv1” on page 130](#).

IKEv1 Utilities and Files

The following table summarizes the configuration files for IKEv1 policy, the storage locations for IKEv1 keys, and the various commands and services that implement IKEv1. For more about

services, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in [“Managing System Services in Oracle Solaris 11.2 ”](#).

TABLE 12-2 IKEv1 Service Name, Commands, Configuration and Key Storage Locations, and Hardware Devices

Service, Command, File, or Device	Description	Man Page
<code>svc:/network/ipsec/ike:default</code>	The SMF service that manages IKEv1.	smf(5)
<code>/usr/lib/inet/in.iked</code>	Internet Key Exchange (IKEv1) daemon. Activates automated key management when the <code>ike</code> service is enabled.	in.iked(1M)
<code>/usr/sbin/ikeadm [-v1]</code>	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)
<code>/usr/sbin/ikecert</code>	Certificate database management command for manipulating local databases that hold public key certificates. The databases can also be stored on attached hardware.	ikecert(1M)
<code>/etc/inet/ike/config</code>	Default configuration file for the IKEv1 policy. Contains the site's rules for matching inbound IKEv1 requests and preparing outbound IKEv1 requests. If this file exists, the <code>in.iked</code> daemon starts when the <code>ike</code> service is enabled. You can change the location of this file by using the <code>svccfg</code> command.	ike.config(4)
<code>ike.preshared</code>	Preshared keys file in the <code>/etc/inet/secret</code> directory. Contains secret keys for authentication in the Phase 1 exchange. Used when configuring IKEv1 with preshared keys.	ike.preshared(4)
<code>ike.privatekeys</code>	Private keys directory in the <code>/etc/inet/secret</code> directory. Contains the private keys that are part of a public-private key pair.	ikecert(1M)
<code>publickeys directory</code>	Directory in the <code>/etc/inet/ike</code> directory that holds public keys and certificate files. Contains the public key part of a public-private key pair.	ikecert(1M)
<code>crls directory</code>	Directory in the <code>/etc/inet/ike</code> directory that holds revocation lists for public keys and certificate files.	ikecert(1M)
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)

IKEv1 Service

The Service Management Facility (SMF) provides the `svc:/network/ipsec/ike:default` service to manage IKEv1. By default, this service is disabled. Before enabling this service, you must create an IKEv1 configuration file, `/etc/inet/ike/config`.

The following `ike` service properties are configurable:

- `config_file` **property** – Sets the location of the IKEv1 configuration file. The initial value is `/etc/inet/ike/config`.
- `debug_level` **property** – Sets the debugging level of the `in.iked` daemon. The initial value is `op`, or operational. For possible values, see the table on debug levels under Object Types in the [ikeadm\(1M\)](#) man page.
- `admin_privilege` **property** – Sets the level of privilege of the `in.iked` daemon. The initial value is `base`. Other values are `modkeys` and `keymat`. For details, see “[IKEv1 ikeadm Command](#)” on page 223.

For information about SMF, see [Chapter 1, “Introduction to the Service Management Facility,”](#) in “[Managing System Services in Oracle Solaris 11.2](#)”. Also see the [smf\(5\)](#), [svcadm\(1M\)](#), and [svccfg\(1M\)](#) man pages.

IKEv1 Daemon

The `in.iked` daemon automates the management of IPsec SAs, which include the cryptographic keys that protect the packets that use IPsec. The daemon securely negotiates ISAKMP SAs and IPsec SAs with a peer system that is running the IKEv1 protocol.

By default, the `svc:/network/ipsec/ike:default` service is not enabled. After you have configured the `/etc/inet/ike/config` file and enabled the `ike:default` service, SMF starts the `in.iked` daemon at system boot. In addition to the `/etc/inet/ike/config` file, further configuration is stored in other files and databases, or as SMF properties. For more information, see “[IKEv1 Utilities and Files](#)” on page 220, and the [ike.preshared\(4\)](#), [ikecert\(1M\)](#), and [in.iked\(1M\)](#) man pages.

After the `ike:default` service is enabled, the `in.iked` daemon reads the configuration files and listens for external requests from an IKE peer and internal requests from IPsec for SAs.

For external requests from an IKEv1 peer, the configuration of the `ike:default` service determines how the daemon responds. Internal requests are routed through the `PF_KEY` interface. This interface handles communication between the kernel part of IPsec, which stores the IPsec SAs and performs packet encryption and decryption, and the key management daemon, `in.iked`, which runs in userland. When the kernel needs an SA to protect a packet, it sends a message through the `PF_KEY` interface to the `in.iked` daemon. For more information, see the [pf_key\(7P\)](#) man page.

Two commands support the IKEv1 daemon. The `ikeadm` command provides a command line interface to the running daemon. The `ikecert` command manages the certificate databases, `ike.privatekeys` and `publickeys`, on your disk and on hardware.

For more information about these commands, see the [in.iked\(1M\)](#), [ikeadm\(1M\)](#), and [ikecert\(1M\)](#) man pages.

IKEv1 Configuration File

The IKEv1 configuration file, `/etc/inet/ike/config`, manages the SAs for network packets that need IPsec protection according to the policies in the IPsec configuration file, `/etc/inet/ipsecinit.conf`.

Key management with IKE includes rules and global parameters. An IKEv1 rule identifies systems that are running another IKEv1 daemon. The rule also specifies the authentication method. Global parameters include such items as the path to an attached hardware accelerator. For examples of IKEv1 policy files, see “[Configuring IKEv2 With Preshared Keys](#)” on page 136. For examples and descriptions of IKEv1 policy entries, see the [ike.config\(4\)](#) man page.

The `/etc/inet/ike/config` file can include the path to a library that is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). IKEv1 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 213.

IKEv1 `ikeadm` Command

You can use the `ikeadm` command to do the following:

- View aspects of the IKE state
- Change the properties of the IKE daemon
- Display statistics on SA creation during the Phase 1 exchange
- Debug IKE protocol exchanges
- Display IKE daemon objects, such as all Phase 1 SAs, policy rules, preshared keys, available Diffie-Hellman groups, Phase 1 encryption and authentication algorithms, and the certificate cache

For examples and a full description of this command's options, see the [ikeadm\(1M\)](#) 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 keys. The base level is the default level of privilege.
keymat level	You can view the actual keys with the <code>ikeadm</code> command.
modkeys level	You can remove, change, and add preshared keys.

For a temporary privilege change, you can use the `ikeadm` command. For a permanent change, change the `admin_privilege` property of the `ike` service. For the temporary privilege change, see [“Managing the Running IKE Daemons” on page 208](#).

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. See [“Security Considerations for ipseckey” on page 214](#). For details that are specific to the `ikeadm` command, see the [`ikeadm\(1M\)` man page](#).

IKEv1 Preshared Keys Files

When you create preshared keys manually, the keys are stored in files in the `/etc/inet/secret` directory. The `ike.preshared` file contains the preshared keys for the Phase 1 exchange when you configure a rule in the `ike/config` to use preshared keys. The `ipseckey` file contains the preshared keys that are used to protect IP packets. The files are protected at `0600`. The secret directory is protected at `0700`.

Because the preshared keys are used to authenticate the Phase 1 exchange, the file must be valid before the `in.iked` daemon starts.

For examples of manually managing IPsec keys, see [“How to Manually Create IPsec Keys” on page 117](#).

IKEv1 Public Key Databases and Commands

The `ikecert` command manages the local system's public/private keys, public certificates, and static CRLs databases. You use this command when the IKEv1 configuration file requires public key certificates. Because IKEv1 uses these databases to authenticate the Phase 1 exchange, the databases must be populated before activating the `in.iked` daemon. Three subcommands handle each of the three databases: `certlocal`, `certdb`, and `certldb`.

If the system has an attached Sun Crypto Accelerator 6000 board, the `ikecert` command uses a PKCS #11 library to access the hardware key and certificate storage.

For more information, see the [`ikecert\(1M\)` man page](#). For information about `metaslot` and the `softtoken` keystore, see the [`cryptoadm\(1M\)` man page](#).

IKEv1 `ikecert tokens` Command

The `tokens` argument lists the token IDs that are available. Token IDs enable the `ikecert certlocal` and `ikecert certdb` commands to generate public key certificates and CSRs. The keys and certificates can also be stored on an attached Sun Crypto Accelerator 6000 board. The `ikecert` command uses the PKCS #11 library to access the hardware keystore.

IKEv1 `ikecert certlocal` Command

The `certlocal` subcommand manages the private key database. Options to this subcommand enable you to add, view, and remove private keys. This subcommand also creates either a self-signed certificate or a CSR. The `-ks` option creates a self-signed certificate. The `-kc` option creates a CSR. Keys are stored on the system in the `/etc/inet/secret/ike.privatekeys` directory, 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 12-3 Correspondences Between `ikecert` Options and `ike/config` Entries in IKEv1

<code>ikecert</code> Option	<code>ike/config</code> Entry	Description
<code>-A subject-alternate-name</code>	<code>cert_trust subject-alternate-name</code>	A nickname that uniquely identifies the certificate. Possible values are an IP address, an email address, or a domain name.
<code>-D X.509-distinguished-name</code>	<code>X.509-distinguished-name</code>	The full name of the certificate authority that includes the country (C), organization name (ON), organizational unit (OU), and common name (CN).
<code>-t dsa-sha1 dsa-sha256</code>	<code>auth_method dsa_sig</code>	An authentication method that is slightly slower than RSA .
<code>-t rsa-md5 and</code>	<code>auth_method rsa_sig</code>	An authentication method that is slightly faster than DSA .
<code>-t rsa-sha1 rsa-sha256 rsa-sha384 rsa-sha512</code>		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.
<code>-t rsa-md5 and</code>	<code>auth_method rsa_encrypt</code>	RSA encryption hides identities in IKE from eavesdroppers but requires that the IKE peers know each other's public keys.
<code>-t rsa-sha1 rsa-sha256 rsa-sha384 rsa-sha512</code>		

If you issue a CSR with the `ikecert certlocal -kc` command, you send the output of the command to a certificate authority (CA). If your company runs its own public key infrastructure (PKI), you send the output to your PKI administrator. The CA or your PKI administrator then creates certificates. The certificates that are returned to you are input to the `certdb`

subcommand. The certificate revocation list (CRL) that the CA returns to you is input for the `certldb` subcommand.

IKEv1 `ikecert certdb` Command

The `certdb` subcommand manages the public key database. Options to this subcommand enable you to add, view, and remove certificates and public keys. The command accepts as input certificates that were generated by the `ikecert certlocal -ks` command on a remote system. For the procedure, see [“How to Configure IKEv1 With Self-Signed Public Key Certificates” on page 168](#). This command also accepts the certificate that you receive from a CA as input. For the procedure, see [“How to Configure IKEv1 With Certificates Signed by a CA” on page 173](#).

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.

IKEv1 `ikecert certldb` Command

The `certldb` subcommand manages the certificate revocation list (CRL) database, `/etc/inet/ike/crls`. The CRL database maintains the revocation lists for public keys. Certificates that are no longer valid are on this list. When CAs provide you with a CRL, you can install the CRL in the CRL database with the `ikecert certldb` command. For the procedure, see [“How to Handle Revoked Certificates in IKEv1” on page 181](#).

IKEv1 `/etc/inet/ike/publickeys` Directory

The `/etc/inet/ike/publickeys` directory contains the public part of a public-private key pair and its certificate in files, or *slots*. The directory is protected at `0755`. The `ikecert certdb` command populates the directory. The `-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 CSR that you sent to the CA. You also install a certificate of the CA.

IKEv1 /etc/inet/secret/ike.privatekeys Directory

The `/etc/inet/secret/ike.privatekeys` directory holds private key files that are part of a public-private key pair. The directory is protected at `0700`. The `ikecert certlocal` command populates the `ike.privatekeys` directory. Private keys are not effective until their public key counterparts, self-signed certificates or CAs, are installed. The public key counterparts are stored in the `/etc/inet/ike/publickeys` directory or on supported hardware.

IKEv1 /etc/inet/ike/crls Directory

The `/etc/inet/ike/crls` directory contains certificate revocation list (CRL) files. Each file corresponds to a public certificate file in the `/etc/inet/ike/publickeys` directory. CAs provide the CRLs for their certificates. You can use the `ikecert certldb` command to populate the database.

Network Security Glossary

3DES	See Triple-DES .
AES	Advanced Encryption Standard. A symmetric 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.
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 packets.
bidirectional tunnel	A tunnel that can transmit packets 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.
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.
chain of trust	In X.509 certificates, the assurance from the certificate authority that the certificates from the trust anchor to the user's certificate provide an unbroken chain of authentication.
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.
digital signature	A digital code that is attached to an electronically transmitted message that uniquely identifies the sender.
distinguished name (DN)	A standardized method of using ordinary strings to represent shared information. Distinguished names are used in LDAP and in X.509 certificates, as well as in other technologies. For more information, see A String Representation of Distinguished Names (http://www.ietf.org/rfc/rfc1779.txt) .
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.
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.
dynamic packet filter	See stateful packet filter .
ECDSA	Elliptic Curve Digital Signature Algorithm. A public key algorithm that is based on elliptic curve mathematics. An ECDSA key size is significantly smaller than the size of a DSA public key needed to generate a signature of the same length.
encapsulating security payload (ESP)	An extension header that provides integrity and confidentiality to packets. 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.
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).
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.
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.

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 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 packet	A packet of information that is carried over IP. An IP packet contains a header and data. The header includes the addresses of the source and the destination of the packet. Other fields in the header help identify and recombine the data with accompanying packets at the destination.
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.
IPsec	IP security. The security architecture that provides protection for IP packets.
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.
label	1. An IKEv2 rule's keyword whose value must match the value of the <code>label</code> keyword in a preshared key file if the <code>auth_method</code> is preshared.

2. A keyword used when creating an IKEv2 certificate. This value is convenient for locating all parts of the certificate (private key, public key, and public key certificate) in the keystore.

3. A mandatory access control (MAC) indication of the level of sensitivity of an object or process. Confidential and Top Secret are sample labels. Labeled network transmissions contain MAC labels.

4. An IKEv1 rule's keyword whose value is used to get the rule.

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.

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 packet with a user priority value. The user priority value indicates how packets 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.

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 \(NAT\)](#).

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.

packet	See IP packet .
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.
perfect forward secrecy (PFS)	<p>In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys. Therefore, PFS can prevent the decryption of previously recorded traffic.</p> <p>PFS applies to authenticated key exchange only. See also Diffie-Hellman algorithm.</p>
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.
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.
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.
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.
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.
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.
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.
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.
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.
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 does not match a request.
stream control transport protocol (SCTP)	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. AES is one example of a symmetric key.
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.
trust anchor	In X.509 certificates, the root certificate from the certificate authority. The certificates from the root certificate to the end certificate establish a chain of trust.
tunnel	<p>The path that is followed by a packet while it is encapsulated. See encapsulation.</p> <p>In IPsec, a configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet.</p>
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 <i>internal</i> 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

Numbers and Symbols

- /etc/inet/hosts file, 103
- /etc/inet/ike/config file
 - cert_root keyword, 174, 180
 - cert_trust keyword, 171, 179
 - description, 133, 223
 - ignore_crls keyword, 176
 - ikecert command and, 225
 - ldap-list keyword, 183
 - pkcs11_path keyword, 178, 224
 - PKCS #11 library entry, 224
 - preshared keys, 162
 - proxy keyword, 183
 - public key certificates, 174, 180
 - putting certificates on hardware, 179
 - sample, 162
 - security considerations, 223
 - self-signed certificates, 171
 - summary, 221
 - use_http keyword, 183
- /etc/inet/ike/crls directory, 227
- /etc/inet/ike/ikev2.config file
 - description, 131, 219
 - preshared keys, 136
 - putting certificates on hardware, 159
 - security considerations, 219
 - self-signed certificates, 145
 - summary, 217
- /etc/inet/ike/ikev2.preshared file
 - description, 219
 - sample, 140
 - summary, 217
 - troubleshooting, 199
 - use, 137, 139
- /etc/inet/ike/kmf-policy.xml file
 - default CA policy, 154
 - definition, 132
 - use, 153, 205
- /etc/inet/ike/publickeys directory, 226
- /etc/inet/ipsecinit.conf file, 212
 - bypassing LAN, 113
 - description, 99
 - location and scope, 98
 - protecting web server, 107
 - purpose, 93
 - sample, 213
 - security considerations, 213
 - tunnel syntax, 109
 - verifying syntax, 104, 114
- /etc/inet/secret/ file, 224
- /etc/inet/secret/ike.preshared file
 - definition, 134
 - sample, 166
 - use, 163, 207
- /etc/inet/secret/ike.privatekeys directory, 227
- /etc/inet/secret/ipseckeys file
 - default path, 212
 - definition, 90
 - storing IPsec keys, 99
 - use, 118, 206
 - verifying syntax, 119
- /var/user/ikeuser, 141

A

- A option
 - dlstat command, 21
 - ikecert certlocal command, 168
- a option
 - digest command, 147

- dladm create-iptun command, 114
- ikecert certdb command, 170, 174
- ikecert certlocal command, 178
- ikecert certrldb command, 183
- ikecert command, 178
- ipadm create-addr command, 114
- ipf command, 63, 66
- ipmon command, 78, 78
- A option
 - ikecert certlocal command, 169
 - ikecert command, 225
- accelerating
 - IKEv1 computations, 191
 - rule processing in IP Filter, 45
 - web server communications, 31
- activating a different rule set
 - packet filtering, 63
- active rule sets *See* IP Filter
- adding
 - CA certificates (IKEv1), 173
 - CA certificates (IKEv2), 151
 - IPsec SAs, 104, 117
 - keys manually (IPsec), 117
 - network management role, 120
 - preshared keys (IKEv1), 165
 - preshared keys (IKEv2), 139
 - public key certificates (IKEv1), 173
 - public key certificates (IKEv2), 151
 - public key certificates (SSL), 36
 - self-signed certificates (IKEv1), 168
 - self-signed certificates (IKEv2), 145
- address pools
 - appending, 72
 - configuration file in IP Filter, 50
 - configuring in IP Filter, 50
 - in IP Filter, 50
 - removing, 71
 - viewing, 71
 - viewing statistics, 76
- AH *See* authentication header (AH)
- Apache web servers
 - accelerating SSL packets, 31
 - configuring with SSL kernel proxy, 33
 - configuring with SSL protection in a zone, 39
 - fallback SSL protection, 36

- SSL kernel proxy and, 33
- SSL kernel proxy and fallback, 36

authentication algorithms

- IKEv1 certificates, 225
- IKEv2 certificates, 149

authentication header (AH)

- compared with ESP, 90, 90
- IPsec protection protocol, 90
- protecting IP packets, 85, 91
- security considerations, 92

B

BPDU protection

- link protection, 16

bypassing

- IPsec on LAN, 113
- IPsec policy, 93

C

-C option

- ksslcfg command, 34

-c option

- in.iked daemon, 163
- in.ikev2d daemon, 137

cert_root keyword

- IKEv1 configuration file, 174, 180

cert_trust keyword

- ikecert command and, 225
- IKEv1 configuration file, 171, 179

certificate authority (CA), 85

- See also* certificates, CSRs
- IKE certificates, 128

certificate revocation lists *See* CRLs

certificate signing requests *See* CSRs

certificate validation policy

- configuring in IKEv2, 153

certificates

- description, 152
- determining if revoked (IKEv2), 155
- dynamic retrieval of revoked, 156
- IKE overview of, 128
- IKEv1
 - adding to database, 174

- CA on hardware, 181
- creating self-signed, 168
- from CA, 174
- ignoring CRLs, 176
- in `ike/config` file, 179
- listing, 170
- requesting from CA, 173
- requesting on hardware, 178
- revoked, 181
- storing, 226
- storing on computer, 166
- storing on hardware, 191
- validating, 170
- verifying, 170
- IKEv2
 - adding to keystore, 152
 - configuring, 154
 - creating self-signed, 145
 - exporting, 147
 - from CA, 152
 - importing, 152
 - in `ikev2.config` file, 159
 - listing, 147
 - policy, 132
 - requesting from CA, 151
 - requesting on hardware, 158
 - revoked, 155
 - storing, 144
 - storing on hardware, 157
 - validating, 147
 - validating certificate policy, 153
 - verifying, 147
- revoking in IKE, 129
- SSL use, 33
- static CRL, 156
- troubleshooting in IKE, 195
- using in IKE, 128
- verifying in IKE, 195
- changing
 - running IKE daemon, 208
- ciphers *See* encryption algorithms
- commands
 - IKEv1
 - description, 224
 - `ikeadm` command, 222, 223
 - `ikecert` command, 221, 222, 224
 - `in.iked` daemon, 222
 - IKEv2
 - description, 220
 - `ikeadm` command, 217, 218, 219, 221
 - `ikev2cert` command, 217, 218, 220
 - `in.ikev2d` daemon, 218
 - IPsec
 - `in.iked` command, 216
 - `ipsecalgs` command, 214
 - `ipseconf` command, 99, 212
 - `ipseckey` command, 89, 99, 214
 - `kstat` command, 215
 - list of, 99
 - security considerations, 214
 - `snoop` command, 215
- computations
 - accelerating IKEv1 in hardware, 191
- configuration files
 - `/etc/inet/secret/ike.preshared`, 134, 163, 166
 - `/etc/inet/secret/ipseckey`s, 90, 118, 212
 - `ike.preshared`, 207
 - `ike/config` file, 221, 223
 - `ike/ikev2.config` file, 217, 219
 - `ike/ikev2.preshared` file, 217
- IP Filter, 46
- IP Filter samples, 80
- configuring
 - address pools in IP Filter, 50
 - Apache 2.2 web server with fallback SSL, 36
 - Apache 2.2 web server with SSL kernel proxy, 33
 - Apache 2.2 web server with SSL protection, 39
- IKEv1
 - CA certificates, 173
 - certificates on hardware, 178
 - mobile systems, 184
 - public key certificates, 167
 - self-signed certificates, 168
- IKEv2
 - CA certificates, 151
 - certificate validation policy, 153
 - certificates on hardware, 157
 - keystore for public certificates, 141
 - preshared keys, 136
 - public key certificates, 144
 - self-signed certificates, 145

- IPsec, 101
 - ipseccinit.conf file, 212
 - link protection, 17, 23
 - NAT rules in IP Filter, 49
 - network security with a role, 119
 - Oracle iPlanet Web Server with SSL kernel proxy, 35
 - packet filtering rules, 46
 - VPN protected by IPsec, 112
 - web servers with SSL kernel proxy, 31
- Configuring IKEv1 for Mobile Systems (Task Map), 184
- Configuring IKEv1 With Public Key Certificates (Task Map), 167
- Configuring IKEv2 With Public Key Certificates (Task Map), 144
- creating, 85
 - See also* adding
 - certificate signing requests (CSRs), 151, 173
 - IKEv2 keystore, 142
 - IP Filter configuration files, 57
 - IPsec SAs, 104, 117
 - ipseccinit.conf file, 104
 - security-related role, 119
 - self-signed certificates (IKEv1), 168
 - self-signed certificates (IKEv2), 145
- CRLs (certificate revocation lists)
 - accessing from central location, 182
 - configuring in IKEv2, 154
 - description, 129
 - ignoring, 176
 - ike/crls database, 227
 - ikecert certrladb command, 226
 - listing, 155, 182
- Cryptographic Framework
 - IPsec and, 214
- CSRs (certificate signing requests)
 - IKEv1
 - from CA, 173
 - on hardware, 178
 - submitting, 174
 - use, 225
 - IKEv2
 - from CA, 151
 - on hardware, 158
 - SSL use, 36
- D**
- D option
 - ikecert certlocal command, 168, 169, 178
 - ikecert command, 225
- daemons
 - in.iked daemon, 131, 132, 221, 222
 - in.ikev2d daemon, 137, 142, 217, 218
 - in.routed daemon, 24
 - webservd daemon, 36
- databases
 - dbfile argument to kmfcfg command, 132
 - ike.privatekeys database, 225, 227
 - ike/crls database, 226, 227
 - ike/publickeys database, 226, 226
 - IKEv1, 224
 - security associations database (SADB), 216
 - security policy database (SPD), 86
- debug_level property
 - IKEv2, 194, 217
- debugging *See* troubleshooting
- default CA policy
 - kmf-policy.xml file, 154
- DefaultFixed network protocol
 - IKEv1, 161
 - IKEv2, 135
 - IPsec, 101
- DHCP protection
 - link protection, 16
- dhcp-nospoof
 - link protection types, 16
- digital signatures in certificates, 225
- directories
 - /etc/apache2/2.2, 38
 - /etc/inet, 221
 - /etc/inet/ike, 217, 217, 221
 - /etc/inet/publickeys, 226
 - /etc/inet/secret, 221
 - /etc/inet/secret/ike.privatekeys, 225
 - /var/user/ikeuser, 141
 - certificates (IKEv1), 226
 - preshared keys, 219, 224
 - private keys (IKEv1), 225
 - public keys (IKEv1), 226
- directory name (DN)

- for accessing CRLs, 182
- displaying defaults
 - IP Filter, 56
- distinguished name (DN)
 - definition, 167
 - example, 129, 169
 - use, 226
- dladm command
 - IPsec tunnel protection, 112
 - link protection, 17
- DSS authentication algorithm, 225

E

- encapsulating security payload (ESP)
 - compared with AH, 90
 - description, 91
 - IPsec protection protocol, 90
 - protecting IP packets, 85
 - security considerations, 92
- encryption algorithms
 - SSL kernel proxy, 32
- ESP *See* encapsulating security payload (ESP)
- export subcommand
 - ikev2cert command, 147
- exporting
 - certificates in IKEv2, 147

F

- F option
 - ipf command, 63, 66, 68
 - ipmon command, 79
 - ipnat command, 69
- f option
 - in.iked daemon, 163
 - in.ikev2d daemon, 137
 - ipf command, 63, 65, 66
 - ipnat command, 70
 - ippool command, 72
 - ksslcfg command, 33
- files
 - httpd.conf, 38

IKEv1

- crls directory, 221, 227
- ike.preshared file, 221, 224
- ike.privatekeys directory, 221, 227
- ike/config file, 100, 133, 221, 223
- publickeys directory, 221, 226

IKEv2

- ike/ikev2.config file, 100, 131, 217, 219
- ike/ikev2.preshared file, 217, 219

IPsec

- ipsecinit.conf file, 99, 99, 212
- ipseckeys file, 99
- kmf-policy.xml, 132, 153
- rsyslog.conf, 77
- ssl.conf, 36
- syslog.conf, 77

FIPS 140

- IKE, 15, 127, 130
- IPsec and, 101
- IPsec configuration and, 96
- Sun Crypto Accelerator 6000 board, 221
- web server 2048-bit key and, 37

flushing *See* deleting

G

- gencert subcommand
 - ikev2cert command, 158
- gencsr subcommand
 - ikev2cert command, 151

H

hardware

- accelerating IKEv1 computations, 191
- finding attached, 157, 191
- public key certificates, 178
- storing IKEv1 keys, 191
- storing IKEv2 keys, 157

hosts file, 103

HTTP access to CRLs

- use_http keyword, 183
- httpd.conf file, 38

I

- I option
 - ipf command, 68
 - ipfstat command, 63
- i option
 - ipfstat command, 63
 - ksslcfg command, 34
- ignore_crls keyword
 - IKEv1 configuration file, 176
- IKE, 85
 - See also* IKEv1, IKEv2
 - certificates, 128
 - displaying IKE information, 202
 - FIPS 140 mode, 15, 127, 130
 - NAT and, 189
 - preshared keys, 127
 - protocol versions, 125
 - reference, 211
 - RFCs, 215
- ike service
 - description, 212, 216
- ike.preshared file *See* /etc/inet/secret/ike.preshared file
- ike.privatekeys database, 227
- ike/config file *See* /etc/inet/ike/config file
- ike/ikev2.config file *See* /etc/inet/ike/ikev2.config file
- ikeadm command
 - description, 218, 219, 222, 223
 - usage summary, 203, 208
- ikecert certlocal command
 - kc option, 173
 - ks option, 168
- ikecert command
 - a option, 178
 - A option, 225
 - certdb subcommand, 170, 174
 - certrldb subcommand, 183
 - description, 218, 222, 224
 - t option, 225
 - tokens subcommand, 191
 - using on hardware, 178
- ikeuser account, 141
- ikeuser directory, 141
- IKEv1
 - adding self-signed certificates, 168
 - changing privilege level, 224
 - checking if valid configuration, 163
 - command descriptions, 220
 - compared with IKEv2 on Oracle Solaris systems, 130
 - configuration files, 220
 - configuring
 - for mobile systems, 184
 - on hardware, 191
 - overview, 161
 - with CA certificates, 173
 - with preshared keys, 162
 - with public key certificates, 167
 - creating self-signed certificates, 168
 - crls database, 227
 - daemon, 222
 - databases, 224
 - generating CSRs, 173
 - ike.preshared file, 224
 - ike.privatekeys database, 227
 - ikeadm command, 223
 - ikecert certdb command, 174
 - ikecert certrldb command, 183
 - ikecert command, 191, 224
 - implementing, 161
 - in.iked daemon, 222
 - ISAKMP SAs, 132
 - key management, 132
 - mobile systems and, 184
 - NAT and, 187
 - perfect forward secrecy (PFS), 132
 - Phase 1 exchange, 132
 - Phase 2 exchange, 133
 - preshared keys, 134, 134, 163, 166
 - privilege level
 - changing, 224
 - description, 223
 - publickeys database, 226
 - security associations, 222
 - service from SMF, 221
 - SMF service description, 220
 - storage locations for keys, 220
 - using a Sun Crypto Accelerator board, 225, 226

- using Sun Crypto Accelerator 6000 board, 191
- IKEv2
 - adding self-signed certificates, 145
 - checking if valid configuration, 137
 - command descriptions, 217
 - compared with IKEv1 on Oracle Solaris systems, 130
 - configuration files, 217
 - configuring
 - CA certificates, 151
 - keystore for public certificates, 141
 - overview, 135
 - with preshared keys, 136
 - with public key certificates, 144
 - creating self-signed certificates, 145
 - daemon, 218
 - generating certificate signing requests, 151
 - ikeadm command, 219
 - ikev2.preshared file, 219
 - ikev2cert command
 - creating self-signed certificate, 145
 - description, 220
 - importing a certificate, 152
 - tokens subcommand, 157
 - using on hardware, 157, 158
 - implementing, 135
 - in.ikev2d daemon, 218
 - ISAKMP SAs, 133
 - key exchange, 131
 - key management, 131
 - key storage, 220
 - listing hardware tokens, 157
 - policy for public certificates, 153
 - security associations, 218
 - service from SMF, 217
 - SMF service description, 217
 - storage location for keys, 217
 - storing public key certificates, 144
 - using Sun Crypto Accelerator 6000 board, 157
 - validating configuration, 197
 - verifying hardware PIN, 143
- ikev2 service
 - ikeuser account, 141
 - use, 105
- ikev2.preshared file *See* /etc/inet/ike/ikev2.preshared file
- ikev2cert command
 - description, 220
 - gencert subcommand, 158
 - gencsr subcommand, 151
 - import subcommand, 148
 - list subcommand, 146, 150
 - setpin subcommand, 142
- ikev2cert gencert command
 - using on hardware, 158
- ikev2cert import command
 - adding a certificate, 152
 - adding key to keystore, 148
 - applying a label, 148
 - CA certificate, 152
- ikev2cert list command
 - using, 155
- ikev2cert tokens command, 143
- import subcommand
 - ikev2cert command, 148
- in.iked daemon
 - activating, 222
 - c option, 163
 - description, 132
 - f option, 163
- in.ikev2d daemon
 - activating, 218
 - c option, 137
 - description, 131
 - f option, 137
- in.routed daemon, 24
- inactive rule sets *See* IP Filter
- Internet Security Association and Key Management Protocol (ISAKMP) SAs
 - description, 133
 - storage location, 219, 224
- IP Filter
 - address pools
 - appending, 72
 - managing, 71
 - removing, 71
 - viewing, 71
 - address pools and, 50

- address pools configuration file, 50
- configuration files, 46
- configuration tasks, 55
- creating
 - log files, 77
- creating configuration files, 57
- disabling, 61
- disabling packet reassembly, 59
- displaying defaults, 56
- displaying statistics, 73
- enabling, 58
- flushing log buffer, 79
- guidelines for using, 44
- ipf command
 - 6 option, 51
- ipfilter service, 44
- ipfstat command
 - 6 option, 51
- ipmon command
 - IPv6 and, 51
- ippool command, 71
 - IPv6 and, 51
- IPv6, 51
- IPv6 configuration files, 51
- log files, 76
- loopback filtering, 60
- man page summaries, 52
- managing packet filtering rule sets, 62
- NAT and, 49
- NAT configuration file, 49
- NAT rules
 - appending, 70
 - viewing, 69
- overview, 41
- packet filtering overview, 46
- packet processing sequence, 42
- removing
 - NAT rules, 69
- rule set
 - activating different, 63
- rule sets
 - active, 62
 - appending to active, 65
 - appending to inactive, 66, 66
 - inactive, 63
 - removing, 64
 - removing inactive, 68
 - switching between, 67
- rule sets and, 45
- sample configuration files, 80
- saving logged packets to a file, 79
- sources, 42
- statistics, 73
- viewing
 - address pool statistics, 76
 - log files, 78
 - state statistics, 74
 - state tables, 73
 - tunable parameters, 75
 - working with rule sets, 61
- IP Filter service
 - defaults, 56
- IP forwarding
 - in IPv4 VPNs, 113
 - in VPNs, 96
- IP packets
 - protecting with IPsec, 85
- IP protection
 - link protection, 16
- IP security architecture *See* IPsec
- ip-nospoof
 - link protection types, 16
- ipadm command
 - hostmodel parameter, 113
 - strict multihoming, 113
- ipf command, 41
 - See also* IP Filter
 - 6 option, 51
 - append rules from command line, 65
 - F option, 64
 - f option, 66
 - I option, 66
 - options, 63
- ipfilter service, 44
- ipfilter:default service, 56
- ipfstat command, 41, 73
 - See also* IP Filter
 - 6 option, 51
 - i option, 62
 - o option, 62

- options, 63
- ipmon command
 - IPv6 and, 51
 - viewing IP Filter logs, 78
- ipnat command, 41
 - See also* IP Filter
 - append rules from command line, 70
 - l option, 69
- ippool command, 41
 - See also* IP Filter
 - append rules from command line, 72
 - F option, 71
 - IPv6 and, 51
 - l option, 71
- IPsec
 - /etc/hosts file, 103
 - activating, 99
 - adding security associations (SAs), 104, 114
 - algorithm source, 214
 - bypassing, 93, 107
 - commands, list of, 99
 - components, 85
 - configuration files, 99
 - configuring, 93, 212
 - configuring by trusted users, 121
 - creating SAs manually, 117
 - Cryptographic Framework and, 214
 - displaying IPsec information, 201
 - encapsulating data, 91
 - encapsulating security payload (ESP), 90, 91
 - extensions to utilities
 - snoop command, 215
 - FIPS 140 and, 96, 101
 - flow chart, 86
 - implementing, 102
 - in.iked daemon, 216
 - in.ikev2d daemon, 216
 - inbound packet process, 86
 - ipsecalgs command, 214
 - ipsecconf command, 93, 212
 - ipsecinit.conf file
 - bypassing LAN, 113
 - configuring, 104
 - description, 212
 - policy file, 93
 - protecting web server, 107
 - tunnel syntax examples, 109
 - ipseckey command, 89, 214
 - IPv4 VPNs, and, 112
 - key management
 - IKEv1, 132
 - IKEv2, 131
 - reference, 216
 - key management
 - ipseckey command, 89
 - kstat command, 215
 - labeled packets and, 102
 - manual key command, 214
 - manual key management, 212
 - manual keys, 90, 118
 - NAT and, 97
 - outbound packet process, 86
 - overview, 85
 - policy command
 - ipsecconf, 212
 - policy files, 212
 - protecting
 - mobile systems, 184
 - packets, 85
 - VPNs, 112
 - web servers, 106
 - protecting a VPN, 108
 - protection policy, 93
 - protection protocols, 90
 - RBAC and, 101
 - RFCs, 215
 - route command, 116
 - running in FIPS 140 mode, 106
 - SCTP protocol and, 98, 102
 - securing traffic, 102
 - security associations (SAs), 86, 89
 - security associations database (SADB), 86, 216
 - security parameter index (SPI), 89
 - security policy database (SPD), 86, 212
 - security protocols, 85, 89
 - security roles, 119
 - services
 - ipsecalgs, 100
 - list of, 99
 - manual-key, 99

- policy, 99
 - summary, 211
 - setting policy
 - permanently, 212
 - temporarily, 212
 - snoop command, 215
 - statistics command, 215
 - transport mode, 93
 - Trusted Extensions labels and, 102
 - tunnel mode, 93
 - tunnels, 95
 - using ssh for secure remote login, 105
 - verifying packet protection, 123
 - virtual machines and, 98
 - virtual private networks (VPNs), 96, 112
 - zones and, 98, 101
- ipsecalgs service, 211
- ipsecconf command
 - configuring IPsec policy, 212
 - description, 99
 - displaying IPsec policy, 106
 - purpose, 93
 - security considerations, 213
 - setting tunnels, 94
 - viewing IPsec policy, 212
- ipsecinit.conf file *See* /etc/inet/ipsecinit.conf file
- ipseckey command
 - description, 89, 99
 - purpose, 214
 - security considerations, 214
- ipseckey file *See* /etc/inet/secret/ipseckey file
- IPv6
 - and IP Filter, 51
- IPv6 in IP Filter
 - configuration files, 51
- K**
- kc option
 - ikecert certlocal command, 173, 225
- kernel
 - accelerating SSL packets, 31
 - SSL kernel proxy for web servers, 31
- key management
 - automatic, 131, 131, 132, 132
 - ike:default service, 216
 - IKEv1, 132
 - IKEv2, 131
 - ikev2 service, 217
 - IPsec, 216
 - ipseckey command, 214
 - manual, 89
 - manual -key service, 216
 - zones and, 101
- key storage
 - IKEv1
 - ISAKMP SAs, 224
 - softtoken keystore, 192, 224
 - token IDs from metaslot, 192
 - IKEv2
 - softtoken keystore, 217, 220
 - IPsec SAs, 99
 - SSL kernel proxy, 33
- keys
 - automatic management, 131, 132
 - creating for IPsec SAs, 117
 - ike.privatekeys database, 227
 - ike/publickeys database, 226
 - managing IPsec, 216
 - manual management in IPsec, 89, 117
 - preshared (IKE), 127
 - preshared (IKEv1), 134
 - storing (IKEv1)
 - certificates, 226
 - private, 225
 - public keys, 226
- keystore
 - creating IKEv2, 142
 - initializing for IKEv2, 141
 - storing IKEv2 certificates, 145
 - using in IKE, 128
- keystore name *See* token ID
- kmf-policy.xml file *See* /etc/inet/ike/kmf-policy.xml file
- kmfcfg command, 153
- ks option
 - ikecert certlocal command, 168, 178, 225
- ksslcfg command, 33, 36
- kstat command, 39
 - and IPsec, 215

L

- L option
 - ipseccomf command, 108
- l option
 - ikecert certdb command, 170
 - ikev2cert list command, 147
 - ipnat command, 69
 - ippool command, 71
- L2 frame protection
 - link protection, 16
- label keyword
 - ikev2.config file, 136
 - ikev2.preshared file, 139
 - ikev2cert gencert command, 145, 150
 - ikev2cert import command, 148, 152
 - ikev2cert list command, 155
 - matching rule to preshared key in IKEv2, 198, 198
- ldap-list keyword
 - IKEv1 configuration file, 183
- LDOMs *See* virtual machines
- link protection
 - configuring, 17, 23
 - dladm command, 17
 - overview, 15
 - verifying, 18
- link protection types
 - against spoofing, 16
 - description, 16
- list subcommand
 - ikev2cert command, 146, 150
- listing
 - algorithms (IPsec), 92
 - certificates, 147, 155, 170, 182
 - CRL (IKEv1), 182
 - CRLs, 155
 - hardware (IKEv1), 191
 - hardware tokens, 157, 157, 191, 192
 - IKE daemon information, 203
- local files name service
 - /etc/inet/hosts file, 103
- local preshared key, 198
- log buffer
 - flushing in IP Filter, 79
- log files

- creating for IP Filter, 77
 - in IP Filter, 76
 - viewing for IP Filter, 78
- logged packets
 - saving to a file, 79
- logical domains *See* virtual machines
- loopback filtering
 - enabling in IP Filter, 60

M

- m option
 - ikecert certlocal command, 168, 178
 - ipadm set-ifprop command, 115
 - kstat command, 39
 - roleadd command, 122
- MAC protection
 - link protection, 16
- mac-nospoof
 - link protection types, 16
- machines
 - network tunables, 23
 - protecting communication, 102
 - protecting link level, 15
 - protecting web servers, 31
 - using a firewall, 55
- manual key management
 - creating, 117
 - IPsec, 90, 118, 212
- manual-key service
 - description, 212, 216
 - use, 119
- metaslot
 - key storage, 192
- mobile systems
 - configuring IKEv1 for, 184

N

- NAT
 - configuration file, 49
 - configuring IP Filter rules for, 49
 - limitations with IPsec, 97
 - NAT rules
 - appending, 70

- viewing, 69
- overview in IP Filter, 49
- removing NAT rules, 69
- RFCs, 97
- using IPsec and IKE, 187, 189
- viewing statistics, 75

Network Address Translation (NAT) *See* NAT

Network IPsec Management rights profile, 120

Network Management rights profile, 120

Network Overall Management role, 120

network protocols

- Automatic, 101, 135, 161
- DefaultFixed
 - IKEv1, 161
 - IKEv2, 135
 - IPsec, 101

Network Security rights profile, 119

O

- o option
 - ipfstat command, 63
 - ipmon command, 78
- OCSP
 - description, 129
 - policy, 154, 183
- openssl command, 36
- Oracle iPlanet Web Server
 - accelerating SSL packets, 31
 - configuring with SSL protection, 35
 - SSL kernel proxy and, 35

P

- p option
 - kslcfg command, 34
- packet filtering
 - activating a different rule set, 63
 - appending
 - rules to active set, 65
 - configuring, 46
 - managing rule sets, 62
 - reloading after updating current rule set, 63
 - removing
 - active rule set, 64

- inactive rule set, 68
- switching between rule sets, 67

packets

- disabling reassembly in IP Filter, 59
- inbound process flowchart, 87
- IP, 85
- outbound process flowchart, 88
- protecting
 - inbound packets, 86
 - outbound packets, 86
 - with IKEv1, 132
 - with IPsec, 86, 90
- verifying protection, 123

peer

- adding to IKEv2 configuration, 139
- creating IKEv2 configuration, 136

perfect forward secrecy (PFS), 132

PF_KEY socket interface, 89, 99

PFS *See* perfect forward secrecy (PFS)

pkcs11_path keyword

- description, 224
- using, 178

pkcs11_token/pin property

- definition, 218
- listing, 143
- use, 143

pkcs11_token/uri property

- definition, 218
- use, 159

PKCS #11 library

- in ike/config file, 224

PKI *See* certificate authority (CA)

policy

- certificate validation, 132, 153, 205
- IPsec, 93

policy files

- ike/config file, 100
- ike/ikev2.config file, 100
- ipsecinit.conf file, 212
- kmf-policy.xml, 132
- security considerations, 213

policy service

- description, 211
- use, 104, 114

preshared keys (IKE), 127

preshared keys (IKEv1)
 definition, 134
 description, 134
 replacing, 164
 sample, 166
 storing, 224
 use, 163
 preshared keys (IKEv2)
 configuring, 136
 matching with rule, 198
 replacing, 138
 storing, 219
 private keys
 storing (IKEv1), 225
 protecting
 IPsec traffic, 85
 mobile systems with IPsec, 184
 network traffic with IPsec, 101
 packets between two systems, 102
 VPN with IPsec in tunnel mode, 112
 web server with IPsec, 106
 Protecting Network Traffic With IPsec (Task Map), 102
 protection protocols
 IPsec, 90
 proxy keyword
 IKEv1 configuration file, 183
 public key certificates *See* certificates
 public keys
 storing (IKEv1), 226
 publickeys database, 226

R

RBAC
 IPsec and, 101
 refreshing
 ikev2 service, 143
 policy service, 114
 preshared keys, 138, 164
 system-log service, 77
 reloading after updating current rule set
 packet filtering, 63
 remote preshared key, 198
 replacing preshared keys, 138, 164

Requests for Comments (RFCs)
 IPv6 Jumbograms, 51
 restricted
 link protection types, 17
 revoked certificates *See* CRLs, OCSP
 rights profiles
 Network IPsec Management, 120
 Network Management, 120
 Network Security, 35
 roles
 creating network security role, 119
 network management role, 120
 route command
 IPsec, 116
 routeadm command
 IP forwarding, 113, 113
 RSA encryption algorithm, 225
 rsyslog.conf entry
 creating for IP Filter, 77
 rule sets, 41
 See also IP Filter
 IP Filter, 61
 NAT in IP Filter, 49
 packet filtering, 45
 rules to inactive set
 appending in IP Filter, 66

S

-s option
 ipf command, 67
 ipfstat command, 74
 ipnat command, 75
 ippool command, 76
 SADB *See* security associations database (SADB)
 SAs *See* security associations (SAs)
 SCA6000 board *See* Sun Crypto Accelerator 6000 board
 SCTP protocol
 IPsec and, 102
 limitations with IPsec, 98
 Secure Sockets Layer (SSL) *See* SSL protocol
 security
 IKEv1, 222
 IKEv2, 218

- IPsec, 85
- security associations (SAs)
 - adding IPsec, 104, 114
 - creating manually, 117
 - definition, 86
 - IKEv1, 222
 - IKEv2, 218
 - IPsec, 89, 104, 114
 - IPsec database, 216
 - ISAKMP, 132
 - random number generation, 131, 133
- security associations database (SADB), 86, 216
- security considerations
 - authentication header (AH), 92
 - comparison of AH and ESP, 90
 - encapsulating security payload (ESP), 92
 - ike/config file, 223
 - ike/ikev2.config file, 219
 - ipseconf command, 213
 - ipsecinit.conf file, 213
 - ipseckey command, 214
 - ipseckey file, 118
 - latched sockets, 213
 - preshared keys, 128
 - security protocols, 92
- security parameter index (SPI), 89
- security policy
 - ike/config file, 100
 - ike/ikev2.config file, 100
 - IPsec, 93
 - ipsecinit.conf file, 212
 - kmf-policy.xml file, 205
- security policy database (SPD), 86, 212
- security protocols
 - authentication header (AH), 91
 - encapsulating security payload (ESP), 91
 - IPsec protection protocols, 90
 - overview, 85
 - Secure Sockets Layer (SSL), 31
 - security considerations, 92
- self-signed certificates
 - configuring in IKEv1, 168
 - configuring in IKEv2, 145
 - IKE overview of, 128
- Service Management Facility (SMF)
 - Apache web server service, 34
 - IKE services, 216
 - IKEv1 service
 - configurable properties, 222
 - description, 221
 - enabling, 187, 222
 - ike service, 221
 - IKEv2 service
 - configurable properties, 217
 - description, 217
 - enabling, 105, 218
 - ike:ikev2 service, 217
 - refreshing, 105
 - IP Filter service
 - checking, 56
 - configuring, 57
 - IPsec services, 211
 - ipsecalgs service, 214
 - list of, 99
 - manual-key description, 216
 - manual-key use, 119, 119
 - policy service, 99
 - SSL kernel proxy service, 34
 - system-log service, 77
- setpin subcommand
 - ikev2cert command, 142
- slots
 - in hardware, 226
- snoop command
 - verifying packet protection, 123
 - viewing protected packets, 215
- sockets
 - IPsec security, 213
- softtoken keystore
 - IKEv2 key storage, 220
 - key storage with metaslot, 192, 224
- spoofing
 - protecting links, 15
- SSL kernel proxy
 - Apache web servers and, 33, 36
 - fall back to Apache web server, 36
 - key storage, 36
 - passphrase files, 36
 - protecting Apache web server in a zone, 39
 - protecting Oracle iPlanet Web Server, 35

- SSL protocol, 31
 - See also* SSL kernel proxy
 - accelerating web servers, 31
 - managing with SMF, 34
 - ssl.conf file, 36
 - state statistics
 - viewing in IP Filter, 74
 - state tables
 - viewing in IP Filter, 73
 - storing
 - certificates on disk, 146
 - certificates on hardware, 157
 - IKEv1 keys on disk, 226, 226
 - keys on disk, 174
 - keys on hardware, 191
 - Sun Crypto Accelerator 6000 board
 - FIPS 140-validated, 221
 - using with IKEv1, 178, 191
 - using with IKEv2, 157
 - syslog.conf entry
 - creating for IP Filter, 77
 - system-log service, 77
 - systems
 - protecting communication, 102
- T**
- T option
 - dladm create-iptun command, 114
 - ikecert certlocal command, 178
 - ikecert command, 226
 - ipadm create-addr command, 114
 - ipf command, 75
 - ksslcfg command, 34
 - t option
 - ikecert certlocal command, 169
 - ikecert command, 225
 - ipfstat command, 73
 - T option
 - ikecert command, 178
 - task maps
 - Configuring IKEv1 for Mobile Systems (Task Map), 184
 - Configuring IKEv1 With Public Key Certificates (Task Map), 167
 - Configuring IKEv2 With Public Key Certificates (Task Map), 144
 - Protecting Network Traffic With IPsec (Task Map), 102
 - TCP/IP networks
 - protecting with ESP, 91
 - token ID
 - in hardware, 226
 - tokens argument
 - ikecert command, 225
 - tokens subcommand
 - ikecert command, 191
 - ikev2cert command, 157
 - transport mode
 - IPsec, 93
 - protected data with ESP, 95
 - troubleshooting
 - IKEv1 payload, 177
 - IP Filter rule sets, 65, 67
 - IPsec and IKE before systems are running, 195
 - IPsec and its key management, 193
 - maintaining current CRLs, 205
 - preparing IPsec and IKE for, 194
 - rights required in IPsec and IKE, 193
 - running IPsec and IKE systems, 196
 - semantic errors in IPsec and IKE, 200
 - Trusted Extensions
 - IPsec and, 102
 - tunable parameters
 - in IP Filter, 75
 - tunnels
 - IPsec, 95
 - modes in IPsec, 93
 - protecting entire inner IP packet, 95
 - protecting packets, 95
 - protecting VPN by using, 112
 - transport mode, 94
 - tunnel keyword in IPsec, 94, 109, 113
 - tunnel mode in IPsec, 93

U

- uniform resource indicator (URI)

- for accessing revoked certificate lists, 182
- use_ftp keyword
 - IKEv1 configuration file, 183
- user
 - managing and configuring IPsec, 120

V

- V option
 - snoop command, 215
- verifying
 - certificate validity (IKEv2), 155
 - hostmodel value, 26
 - IKE certificate by its fingerprint, 150
 - IKE certificates, 128
 - ikev2.config syntax, 137
 - ipsecinit.conf syntax, 104, 114, 114
 - ipseckey syntax, 119
 - link protection, 18
 - packet protection, 123
 - routing daemon disabled, 24
 - self-signed certificate validity, 147
- viewing
 - active IKE rules, 204
 - address pool statistics in IP Filter, 76
 - address pools in IP Filter, 71
 - certificate validation policy, 205
 - IKE information, 202
 - IKE preshared keys, 203
 - IKE property values, 202
 - IKE SAs, 204
 - IP Filter log files, 78
 - IPsec configuration, 212
 - IPsec information, 201
 - manual keys for IPsec information, 201
 - NAT statistics in IP Filter, 75
 - state of IKE daemon, 203
 - state statistics in IP Filter, 74
 - state tables in IP Filter, 73
 - tunable parameters in IP Filter, 75
- virtual machines
 - IPsec and, 98
- virtual private networks (VPNs)
 - configuring with routeadm command, 113, 113
 - constructed with IPsec, 96

- IPv4 example, 112
- protecting with IPsec, 112
- tunnel mode and, 109

VPN *See* virtual private networks (VPNs)

W

- web servers
 - accelerating SSL packets, 31
 - protecting backend communications, 106
 - using SSL kernel proxy, 31
- websrvd daemon, 36
- Wireshark application
 - installing, 194
 - URL, 215
 - using, 196
 - using with snoop command, 124

X

- x option
 - ksslcfg command, 34

Z

- zones
 - configuring Apache web server with SSL protection, 39
 - IPsec and, 98, 101
 - key management and, 101
 - static IP address in IPsec, 98