NAME

iked – Internet Key Exchange version 2 (IKEv2) daemon

SYNOPSIS

iked [**−dnSTv**] [**−D** *macro*=*value*] [**−f** *file*]

DESCRIPTION

iked is an Internet Key Exchange (IKEv2) daemon which performs mutual authentication and which establishes and maintains IPsec flows and security associations (SAs) between the two peers.

The IKEv2 protocol is defined in RFC 4306, which combines and updates the previous standards: ISAKMP/Oakley (RFC 2408), IKE (RFC 2409), and the Internet DOI (RFC 2407). **iked** only supports the IKEv2 protocol; support for ISAKMP/Oakley and IKEv1 is provided by isakmpd(8).

The options are as follows:

−D *macro*=*value*
: Define *macro* to be set to *value* on the command line. Overrides the definition of *macro* in the configuration file.

−d
: Do not daemonize and log to *stderr*.

−f *file*
: Use *file* as the configuration file, instead of the default /etc/iked.conf.

−n
: Configtest mode. Only check the configuration file for validity.

−S
: Start **iked** in passive mode. See the **set passive** option in iked.conf(5) for more information.

−T
: Disable NAT-Traversal and do not propose NAT-Traversal support to the peers.

−v
: Produce more verbose output.

FILES

/etc/iked.conf	The default **iked** configuration file.
/etc/iked/ca/	The directory where CA certificates are kept.
/etc/iked/certs/	The directory where IKE certificates are kept, both the local certificate(s) and those of the peers, if a choice to have them kept permanently has been made.
/etc/iked/crls/	The directory where CRLs are kept.
/etc/iked/private/	The directory where local private keys used for public key authentication are kept. The file local.key is used to store the local private key.
/etc/iked/pubkeys/	The directory in which trusted public keys are kept. The keys must be named in the fashion described above.
/var/run/iked.sock	The default **iked** control socket.

SEE ALSO

iked.conf(5), ikectl(8), isakmpd(8)

Internet Key Exchange (IKEv2) Protocol, RFC 4306, December 2005.

HISTORY

The **iked** program first appeared in OpenBSD 4.8.

AUTHORS

The **iked** program was written by Reyk Floeter <reyk@vantronix.net>.

CAVEATS

iked is not yet finished and is missing some important security features. It should not yet be used in production networks.

NAME

iked.conf – IKEv2 configuration file

DESCRIPTION

iked.conf is the configuration file for iked(8), the Internet Key Exchange version 2 (IKEv2) daemon for IPsec. IPsec itself is a pair of protocols: Encapsulating Security Payload (ESP), which provides integrity and confidentiality; and Authentication Header (AH), which provides integrity. The IPsec protocol itself is described in ipsec(4).

In its most basic form, a flow is established between hosts and/or networks, and then Security Associations (SA) are established, which detail how the desired protection will be achieved. IPsec uses flows to determine whether to apply security services to an IP packet or not. iked(8) is used to set up flows and establish SAs automatically, by specifying 'ikev2' policies in **iked.conf** (see AUTOMATIC KEYING POLICIES, below).

Alternative methods of setting up flows and SAs are also possible using manual keying or automatic keying using the older ISAKMP/Oakley a.k.a. IKEv1 protocol. Manual keying is not recommended, but can be convenient for quick setups and testing. See ipsec.conf(5) and isakmpd(8) for more information about manual keying and ISAKMP support.

IKED.CONF FILE FORMAT

iked.conf is divided into three main sections:

Macros
User-defined variables may be defined and used later, simplifying the configuration file.

Global Configuration
Global settings for iked(8).

Automatic Keying Policies
Policies to set up IPsec flows and SAs automatically.

Lines beginning with '#' and empty lines are regarded as comments, and ignored. Lines may be split using the '\' character.

Addresses can be specified in CIDR notation (matching netblocks), as symbolic host names, interface names, or interface group names.

Additional configuration files can be included with the **include** keyword, for example:

```
include "/etc/macros.conf"
```

MACROS

Macros can be defined that will later be expanded in context. Macro names must start with a letter, and may contain letters, digits and underscores. Macro names may not be reserved words (for example **flow**, **from**, **esp**). Macros are not expanded inside quotes.

For example:

```
remote_gw = "192.168.3.12"
ikev2 esp from 192.168.7.0/24 to 192.168.8.0/24 peer $remote_gw
```

GLOBAL CONFIGURATION

Here are the settings that can be set globally:

set active
Set `iked`(8) to active mode. This is the default.

set passive
Set `iked`(8) to passive mode. In passive mode no packets are sent to peers and no connections are initiated by `iked`(8). This option is used for setups using `sasyncd`(8) and `carp`(4) to provide redundancy. iked will run in passive mode until sasyncd has determined that the host is the master and can switch to active mode.

set couple
Load the negotiated security associations (SAs) and flows into the kernel. This is the default.

set decouple
Don't load the negotiated SAs and flows from the kernel. This mode is only useful for testing and debugging.

user *name password*
`iked`(8) supports user-based authentication by tunneling the Extensible Authentication Protocol (EAP) over IKEv2. In its most basic form, the users will be authenticated against a local, integrated password database that is configured with the **user** lines in **iked.conf** and the *name* and *password* arguments. Note that the password has to be specified in plain text which is required to support different challenge-based EAP methods like EAP-MD5 or EAP-MSCHAPv2.

AUTOMATIC KEYING POLICIES

This section is used to configure policies that will be used by `iked`(8) to set up flows and SAs automatically. Some examples of setting up automatic keying:

```
# Set up a VPN:
# First between the gateway machines 192.168.3.1 and 192.168.3.2
# Second between the networks 10.1.1.0/24 and 10.1.2.0/24
ikev2 esp from 192.168.3.1 to 192.168.3.2
ikev2 esp from 10.1.1.0/24 to 10.1.2.0/24 peer 192.168.3.2
```

The commands are as follows:

ikev2 [*name*] [*mode*] [*encap*]
name is an optional arbitrary string identifying the policy. The name should only occur once in **iked.conf** or any included files. If omitted, a name will be generated automatically for the policy.

mode specifies the IKEv2 mode to use: one of *passive* or *active*. When *passive* is specified, `iked`(8) will not immediately start negotiation of this tunnel, but wait for an incoming request from the remote peer. When *active* is specified, negotiation will be started at once. If omitted, *passive* mode will be used.

encap specifies the encapsulation protocol to be used. Possible protocols are *esp* and *ah*; the default is *esp*.

proto *protocol*
The optional **proto** parameter restricts the flow to a specific IP protocol. Common protocols are `icmp`(4), `tcp`(4), and `udp`(4). For a list of all the protocol name to number mappings used by `iked`(8), see the file `/etc/protocols`.

from *src* [**port** *sport*] [(*srcnat*)] **to** *dst* [**port** *dport*]
Specify one or more traffic selectors for this policy which will be used to negotiate the IPsec flows between the IKEv2 peers. During the negotiation, the peers may decide to narrow a flow to a subset of the configured traffic selector networks to match the policies on each side.

Each traffic selector will apply for packets with source address `src` and destination address `dst`. The keyword `any` will match any address (i.e. 0.0.0.0/0). If the `src` argument specifies a fictional source ID, the `srcnat` parameter can be used to specify the actual source address. This can be used in outgoing NAT/BINAT scenarios as described below.

The optional **port** modifiers restrict the traffic selectors to the specified ports. They are only valid in conjunction with the `tcp`(4) and `udp`(4) protocols. Ports can be specified by number or by name. For a list of all port name to number mappings used by `ipsecctl`(8), see the file `/etc/services`.

local *localip* **peer** *remote*

The **local** parameter specifies the address or FQDN of the local endpoint. Unless the gateway is multi-homed or uses address aliases, this option is generally not needed.

The **peer** parameter specifies the address or FQDN of the remote endpoint. For host-to-host connections where *dst* is identical to *remote*, this option is generally not needed as it will be set to *dst* automatically. If it is not specified or if the keyword *any* is given, the default peer is used.

ikesa auth *algorithm* **enc** *algorithm* **prf** *algorithm* **group** *group*

These parameters define the mode and cryptographic transforms to be used for the IKE SA negotiation, also known as phase 1. The IKE SA will be used to authenticate the machines and to set up an encrypted channel for the IKEv2 protocol.

Possible values for **auth**, **enc**, **prf**, **group**, and the default proposals are described below in CRYPTO TRANSFORMS. If omitted, `iked`(8) will use the default proposals for the IKEv2 protocol.

childsa auth *algorithm* **enc** *algorithm* **group** *group*

These parameters define the cryptographic transforms to be used for the Child SA negotiation, also known as phase 2. Each Child SA will be used to negotiate the actual IPsec SAs. The initial Child SA is always negotiated with the initial IKEv2 key exchange; additional Child SAs may be negotiated with additional Child SA key exchanges for an established IKE SA.

Possible values for **auth**, **enc**, **group**, and the default proposals are described below in CRYPTO TRANSFORMS. If omitted, `iked`(8) will use the default proposals for the ESP or AH protocol. The **group** option will only be used to enable Perfect Forwarding Security (PFS) for additional Child SAs exchanges that are not part of the initial key exchange.

srcid *string* **dstid** *string*

srcid defines an ID of type "FQDN", "DER_ASN1_DN", "IPV4_ADDR", "IPV6_ADDR", or "RFC822_ADDR" that will be used by `iked`(8) as the identity of the local peer. If the argument is an email address (reyk@example.com), `iked`(8) will use RFC822_ADDR as the ID type. The DER_ASN1_DN type will be used if the string starts with a slash '/' (/C=DE/../CN=10.0.0.1/emailAddress=reyk@example.com). If the argument is an IPv4 address or a compressed IPv6 address, the ID types IPV4_ADDR or IPV6_ADDR will be used. Anything else is considered to be an FQDN.

If **srcid** is omitted, the default is to use the hostname of the local machine, see `hostname`(1) to set or print the hostname.

dstid is similar to **srcid**, but instead specifies the ID to be used by the remote peer.

[*ikeauth*]

Specify the mode to mutually authenticate the peers. Non-psk modes will require to set up certificates and RSA public keys; see `iked`(8) for more information.

eap *type*
: Use EAP to authenticate the initiator. The only supported EAP *type* is currently *MSCHAP-V2*. The responder will use RSA public key authentication.

psk *string*
: Use a pre-shared key *string* or hex value (starting with 0x) for authentication.

rsa
: Use RSA public key authentication. This is the default mode if no option is specified.

tag *string*

Add a pf(4) tag to all packets of IPsec SAs created for this connection. This will allow matching packets for this connection by defining rules in pf.conf(5) using the **tagged** keyword.

The following variables can be used in tags to include information from the remote peer on runtime:

$id
: The **dstid** that was proposed by the remote peer to identify itself. It will be expanded to *id-value*, e.g. *foo.example.com*. To limit the size of the derived tag, iked(8) will extract the common name 'CN=' from DER_ASN1_DN IDs, for example */C=DE/../CN=10.1.1.1/..* will be expanded to *10.1.1.1*.

$domain
: Extract the domain from IDs of type FQDN, RFC822_ADDR or DER_ASN1_DN.

$name
: The name of the IKEv2 policy that was configured in **iked.conf** or automatically generated by iked(8).

For example, if the ID is *foo.example.com* or *user@example.com*, "ipsec-$domain" expands to "ipsec-example.com". The variable expansion for the *tag* directive occurs only at runtime, not during configuration file parse time.

PACKET FILTERING

IPsec traffic appears unencrypted on the enc(4) interface and can be filtered accordingly using the OpenBSD packet filter, pf(4). The grammar for the packet filter is described in pf.conf(5).

The following components are relevant to filtering IPsec traffic:

external interface
Interface for IKE traffic and encapsulated IPsec traffic.

proto udp port 500
IKE traffic on the external interface.

proto udp port 4500
IKE NAT-Traversal traffic on the external interface.

proto ah | esp
Encapsulated IPsec traffic on the external interface.

enc0
Interface for outgoing traffic before it's been encapsulated, and incoming traffic after it's been decapsulated. State on this interface should be interface bound; see enc(4) for further information.

proto ipencap
[tunnel mode only] IP-in-IP traffic flowing between gateways on the enc0 interface.

tagged ipsec-example.org
Match traffic of IPsec SAs using the **tag** keyword.

If the filtering rules specify to block everything by default, the following rule would ensure that IPsec traffic never hits the packet filtering engine, and is therefore passed:

```
set skip on enc0
```

In the following example, all traffic is blocked by default. IPsec-related traffic from gateways {192.168.3.1, 192.168.3.2} and networks {10.0.1.0/24, 10.0.2.0/24} is permitted.

```
block on ix0
block on enc0

pass  in on ix0 proto udp from 192.168.3.2 to 192.168.3.1 \
        port {500, 4500}
pass out on ix0 proto udp from 192.168.3.1 to 192.168.3.2 \
        port {500, 4500}

pass  in on ix0 proto esp from 192.168.3.2 to 192.168.3.1
pass out on ix0 proto esp from 192.168.3.1 to 192.168.3.2

pass  in on enc0 proto ipencap from 192.168.3.2 to 192.168.3.1 \
        keep state (if-bound)
pass out on enc0 proto ipencap from 192.168.3.1 to 192.168.3.2 \
        keep state (if-bound)
pass  in on enc0 from 10.0.2.0/24 to 10.0.1.0/24 \
        keep state (if-bound)
pass out on enc0 from 10.0.1.0/24 to 10.0.2.0/24 \
        keep state (if-bound)
```

pf(4) has the ability to filter IPsec-related packets based on an arbitrary *tag* specified within a ruleset. The tag is used as an internal marker which can be used to identify the packets later on. This could be helpful, for example, in scenarios where users are connecting in from differing IP addresses, or to support queue-based bandwidth control, since the enc0 interface does not support it.

The following pf.conf(5) fragment uses queues for all IPsec traffic with special handling for developers and employees:

```
altq on ix0 cbq bandwidth 1000Mb \
        queue { deflt, developers, employees, ipsec }
    queue deflt bandwidth 10% priority 0 cbq(default ecn)
    queue developers bandwidth 75% priority 7 cbq(borrow red)
    queue employees bandwidth 5% cbq(red)
    queue ipsec bandwidth 10% cbq(red)

pass out on ix0 proto esp queue ipsec

pass out on ix0 tagged ipsec-developers.example.com queue developers
pass out on ix0 tagged ipsec-employees.example.com queue employees
```

The tags will be assigned by the following **iked.conf** example:

```
ikev2 esp from 10.1.1.0/24 to 10.1.2.0/24 peer 192.168.3.2 \
        tag ipsec-$domain
```

OUTGOING NETWORK ADDRESS TRANSLATION

In some network topologies it is desirable to perform NAT on traffic leaving through the VPN tunnel. In order to achieve that, the *src* argument is used to negotiate the desired network ID with the peer and the *srcnat* parameter defines the true local subnet, so that a correct SA can be installed on the local side.

For example, if the local subnet is 192.168.1.0/24 and all the traffic for a specific VPN peer should appear as coming from 10.10.10.1, the following configuration is used:

```
ikev2 esp from 10.10.10.1 (192.168.1.0/24) to 192.168.2.0/24 \
        peer 10.10.20.1
```

Naturally, a relevant NAT rule is required in `pf.conf`(5). For the example above, this would be:

```
match on enc0 from 192.168.1.0/24 to 192.168.2.0/24 nat-to 10.10.10.1
```

From the peer's point of view, the local end of the VPN tunnel is declared to be 10.10.10.1 and all the traffic arrives with that source address.

CRYPTO TRANSFORMS

The following authentication types are permitted with the **`auth`** keyword:

Authentication	*Key Length*	*Truncated Length*
`hmac-md5`	128 bits	96 bits
`hmac-sha1`	160 bits	96 bits
`hmac-sha2-256`		256 bits 128 bits
`hmac-sha2-384`		384 bits 192 bits
`hmac-sha2-512`		512 bits 256 bits

The following pseudo-random function types are permitted with the **`prf`** keyword:

Authentication	*Key Length*	
`hmac-md5`	128 bits	[IKE only]
`hmac-sha1`	160 bits	[IKE only]
`hmac-sha2-256`		256 bits [IKE only]
`hmac-sha2-384`		384 bits [IKE only]
`hmac-sha2-512`		512 bits [IKE only]

The following cipher types are permitted with the **`enc`** keyword:

Cipher	*Key Length*	
`des`	56 bits	[ESP only]
`3des`	168 bits	
`aes-128`	128 bits	
`aes-192`	192 bits	
`aes-256`	256 bits	
`aes-ctr`	160 bits	[ESP only]
`blowfish`	160 bits	[ESP only]
`cast`	128 bits	[ESP only]
`null`		[ESP only]

Use of DES as an encryption algorithm is not recommended (except for backwards compatibility) due to the short key length.

DES requires 8 bytes to form a 56-bit key and 3DES requires 24 bytes to form its 168-bit key. This is because the most significant bit of each byte is used for parity.

The keysize of AES-CTR is actually 128-bit. However as well as the key, a 32-bit nonce has to be supplied. Thus 160 bits of key material have to be supplied.

Using NULL with ESP will only provide authentication. This is useful in setups where AH can not be used, e.g. when NAT is involved.

The following group types are permitted with the **group** keyword:

Name	*Group*	*Size*	*Type*
`modp768`	grp1	768	MODP
`modp1024`	grp2	1024	MODP
`ec155`	grp3	155	EC2N [insecure]
`ec185`	grp4	185	EC2N [insecure]
`modp1536`	grp5	1536	MODP
`modp2048`	grp14	2048	MODP
`modp3072`	grp15	3072	MODP
`modp4096`	grp16	4096	MODP
`modp6144`	grp17	6144	MODP
`modp8192`	grp18	8192	MODP
`ec256`	grp19	256	ECP
`ec384`	grp20	384	ECP
`ec521`	grp21	521	ECP
`modp1024-160`	grp22	2048	MODP, 160 bit Prime Order Subgroup
`modp2048-224`	grp23	2048	MODP, 224 bit Prime Order Subgroup
`modp2048-256`	grp24	2048	MODP, 256 bit Prime Order Subgroup
`ec192`	grp25	192	ECP
`ec224`	grp26	224	ECP

The currently supported group types are either MODP (exponentiation groups modulo a prime), EC2N (elliptic curve groups over GF[2^N]), or ECP (elliptic curve groups modulo a prime). Please note that the EC2N groups are considered as insecure and only provided for backwards compatibility.

EXAMPLES

The first example is intended for clients connecting to `iked`(8) as an IPsec gateway, or IKEv2 responder, using mutual public key authentication and additional challenge-based EAP-MSCHAPv2 password authentication:

```
user "test" "password123"

ikev2 "win7" esp \
        from 172.16.2.0/24 to 0.0.0.0/0 \
        peer 10.0.0.0/8 local 192.168.56.0/24 \
        eap "mschap-v2" \
        config address 172.16.2.1 \
        tag "$name-$id"
```

The next example allows peers to authenticate using a pre-shared key 'foobar':

```
ikev2 "big test" \
        esp proto tcp \
        from 10.0.0.0/8 port 23 to 20.0.0.0/8 port 40 \
        from 192.168.1.1 to 192.168.2.2 \
        peer any local any \
        ikesa enc 3des auth hmac-sha1 group modp1024 \
        childsa enc aes-128 auth hmac-sha1 \
        srcid host.example.com \
        dstid 192.168.0.254 \
        psk "foobar"
```

SEE ALSO

`enc`(4), `ipsec`(4), `ipsec.conf`(5), `pf.conf`(5), `ikectl`(8), `iked`(8)

HISTORY

The **`iked.conf`** file format first appeared in OpenBSD 4.8.

AUTHORS

The **`iked.conf`** program was written by Reyk Floeter <reyk@vantronix.net>.

NAME

ikectl – control the IKEv2 daemon

SYNOPSIS

ikectl [**-s** *socket*] *command* [*arg ...*]

DESCRIPTION

The **ikectl** program controls the iked(8) daemon and provides commands to maintain a simple X.509 certificate authority (CA) for IKEv2 peers.

The options are as follows:

-s *socket*
: Use *socket* instead of the default /var/run/iked.sock to communicate with iked(8).

IKED CONTROL COMMANDS

The following commands are available to control iked(8):

active
: Set iked(8) to active mode.

passive
: Set iked(8) to passive mode. In passive mode no packets are sent to peers and no connections are initiated by iked(8).

couple
: Load the negotiated security associations (SAs) and flows into the kernel.

decouple
: Unload the negotiated SAs and flows from the kernel. This mode is only useful for testing and debugging.

load *filename*
: Reload the configuration from the specified file.

log brief
: Disable verbose logging.

log verbose
: Enable verbose logging.

monitor
: Monitor internal messages of the iked(8) subsystems.

reload
: Reload the configuration from the default configuration file.

reset all
: Reset the the running state.

reset ca
: Reset the X.509 CA and certificate state.

reset policy
: Flush the configured policies.

reset sa
Flush the running SAs.

reset user
Flush the local user database.

PKI AND CERTIFICATE AUTHORITY COMMANDS

In order to use public key based authentication with IKEv2, a public key infrastructure (PKI) has to be set up to create and sign the peer certificates. **ikectl** includes commands to simplify maintenance of the PKI and to set up a simple certificate authority (CA) for iked(8) and its peers.

The following commands are available to control the CA:

ca *name* **create**
Create a new certificate authority with the specified *name*.

ca *name* **delete**
Delete the certificate authority with the specified *name*.

ca *name* **install**
Install the certificate and Certificate Revocation List (CRL) for CA *name* as the currently active CA.

ca *name* **certificate** *host* Cm0
Create a private key and certificate for *host* and sign then with the key of certificate authority with the speicified *name*.

ca *name* **certificate** *host* Cm0
Deletes the private key and and certificates associated with *host*.

ca *name* **certificate** *host* Cm0
Export key files for *host* of the certificate authority with the specified *name* into the current directory for transport to other systems.

ca *name* **certificate** *host* Cm0
Install the private and public key for *host* into the active configuration.

ca *name* **certificate** *host* Cm0
Revoke the certificate specified by *host* and generate a new Certificate Revocation List (CRL).

show ca *name* **certificates**
Display a listing of certificates associated with CA *name*.

ca *name* **key** *host* Cm0
Create a private key for *host* if one does not already exist.

ca *name* **key** *host* Cm0
Install the private and public keys for *host* into the active configuration.

ca *name* **key** *host* Cm0
Delete the private key for *host*.

ca *name* **key** *host* Cm0**0**Cm1**1**Cm2
Source the private key for *host* from the named *file*.

FILES

/etc/ssl/	Directory to store the CA files.
/var/run/iked.sock	default UNIX-domain socket used for communication with `iked`(8)

SEE ALSO

`iked`(8), `ssl`(8)

HISTORY

The **`ikectl`** program first appeared in OpenBSD 4.8.

AUTHORS

The **`ikectl`** program was written by Reyk Floeter <reyk@vantronix.net>.