NAME

wg - WireGuard protocol driver

SYNOPSIS

To load the driver as a module at boot time, place the following line in loader.conf(5):

```
if_wg_load="YES"
```

DESCRIPTION

The **wg** driver provides Virtual Private Network (VPN) interfaces for the secure exchange of layer 3 traffic with other WireGuard peers using the WireGuard protocol.

A **wg** interface recognizes one or more peers, establishes a secure tunnel with each on demand, and tracks each peer's UDP endpoint for exchanging encrypted traffic with.

The interfaces can be created at runtime using the **ifconfig wg***N* **create** command. The interface itself can be configured with wg(8).

The following glossary provides a brief overview of WireGuard terminology:

Peer
: Peers exchange IPv4 or IPv6 traffic over secure tunnels. Each **wg** interface may be configured to recognise one or more peers.

Key
: Each peer uses its private key and corresponding public key to identify itself to others. A peer configures a **wg** interface with its own private key and with the public keys of its peers.

Pre-shared key
: In addition to the public keys, each peer pair may be configured with a unique pre-shared symmetric key. This is used in their handshake to guard against future compromise of the peers' encrypted tunnel if an attack on their Diffie-Hellman exchange becomes feasible. It is optional, but recommended.

Allowed IP addresses
: A single **wg** interface may maintain concurrent tunnels connecting diverse networks. The interface therefore implements rudimentary routing and reverse-path filtering functions for its tunneled traffic. These functions reference a set of allowed IP address ranges configured against each peer.

  The interface will route outbound tunneled traffic to the peer configured with the most specific matching allowed IP address range, or drop it if no such match exists.

The interface will accept tunneled traffic only from the peer configured with the most specific matching allowed IP address range for the incoming traffic, or drop it if no such match exists. That is, tunneled traffic routed to a given peer cannot return through another peer of the same **wg** interface. This ensures that peers cannot spoof one another's traffic.

Handshake
: Two peers handshake to mutually authenticate each other and to establish a shared series of secret ephemeral encryption keys. Either peer may initiate a handshake. Handshakes occur only when there is traffic to send, and recur every two minutes during transfers.

Connectionless
: Due to the handshake behavior, there is no connected or disconnected state.

Keys

Private keys for WireGuard can be generated from any sufficiently secure random source. The Curve25519 keys and the pre-shared keys are both 32 bytes long and are commonly encoded in base64 for ease of use.

Keys can be generated with wg(8) as follows:

```
$ wg genkey
```

Although a valid Curve25519 key must have 5 bits set to specific values, this is done by the interface and so it will accept any random 32-byte base64 string.

EXAMPLES

Create a **wg** interface and set random private key.

```
# ifconfig wg0 create
# wg genkey | wg set wg0 listen-port 54321 private-key /dev/stdin
```

Retrieve the associated public key from a **wg** interface.

```
$ wg show wg0 public-key
```

Connect to a specific endpoint using its public-key and set the allowed IP address

```
# wg set wg0 peer '7lWtsDdqaGB3EY9WNxRN3hVaHMtu1zXw71+bOjNOVUw=' endpoint 10.0.1.100:54321 allo
```

Remove a peer

```
# wg set wg0 peer '7lWtsDdqaGB3EY9WNxRN3hVaHMtu1zXw71+bOjNOVUw=' remove
```

DIAGNOSTICS

The **wg** interface supports runtime debugging, which can be enabled with:

ifconfig wg*N* **debug**

Some common error messages include:

Handshake for peer X did not complete after 5 seconds, retrying Peer X did not reply to our initiation packet, for example because:

- The peer does not have the local interface configured as a peer. Peers must be able to mutually authenticate each other.
- The peer endpoint IP address is incorrectly configured.
- There are firewall rules preventing communication between hosts.

Invalid handshake initiation The incoming handshake packet could not be processed. This is likely due to the local interface not containing the correct public key for the peer.

Invalid initiation MAC The incoming handshake initiation packet had an invalid MAC. This is likely because the initiation sender has the wrong public key for the handshake receiver.

Packet has unallowed src IP from peer X After decryption, an incoming data packet has a source IP address that is not assigned to the allowed IPs of Peer X.

SEE ALSO

inet(4), ip(4), ipsec(4), netintro(4), ovpn(4), ipf(5), pf.conf(5), ifconfig(8), ipfw(8), wg(8)

WireGuard whitepaper, https://www.wireguard.com/papers/wireguard.pdf.

HISTORY

The **wg** device driver first appeared in FreeBSD 13.2.

AUTHORS

The **wg** device driver was written by Jason A. Donenfeld <*Jason@zx2c4.com*>, Matt Dunwoodie <*ncon@nconroy.net*>, Kyle Evans <*kevans@FreeBSD.org*>, and Matt Macy <*mmacy@FreeBSD.org*>.

This manual page was written by Gordon Bergling <*gbe@FreeBSD.org*> and is based on the OpenBSD manual page written by David Gwynne <*dlg@openbsd.org*>.