if_ovpn

OpenVPN Data Channel Offload
OpenVPN

• VPN (No, really!)

• Originally developed by James Yonan

• First release May 13th, 2001

• p2p, client/server

• pre-shared key, certificate or username/password based authentication

• Windows, Linux, macOS, Android, AIX, FreeBSD, OpenBSD, DragonflyBSD, …
The problem

- It’s slow
The problem

- It’s slow
- Because user space
The problem

User space

Application

OpenVPN

Routing

Crypto

Kernel space

if_tun

socket

em0

Internet
What is DCO

• Data Channel Offload
  • i.e. data channel in the kernel
• if_ovpn

• No more (extra) copying between user and kernel space
• Also, hardware accelerators!
  • (Although user space already got AES-NI, which is the one you have)
DCO

User space

Application

OpenVPN

Kernel space

ovpn0

ioctl

DCO configuration

Routing

Crypto

if_ovpn

Internet

netgate
Limitations

- Only AES-GCM or ChaCha20/Poly1305
- No
  - Compression
  - Fragmentation
  - Layer 2
  - Topologies other than subnet
  - Traffic shaping (in OpenVPN)
- OpenVPN used this as an opportunity for a clean break
- Clients must be OpenVPN 2.4 or greater
Considerations

- UDP vs. TCP
- Multiplexing socket
- Locking
- loctl
  - nvlist
- Routing
- Key rotation
- vnet
UDP vs. TCP

- OpenVPN supports tunnels over TCP
- Because firewalls
- FreeBSD’s if_ovpn is UDP-only
- We didn’t care enough about TCP
- No equivalent to `udp_set_kernel_tunneling()` for TCP
Multiplexing

- Control channel (userspace) vs. data channel (kernel)
- Share a single socket
- `fd` passed to kernel during setup
- Kernel passes unknown (i.e. control) packets to userspace
- One of the few network stack modification required
Locking design

- rm_lock
  - Read-lock when processing incoming or outgoing packets
  - Write-lock for configuration changes
- Exceptions
  - Counters
  - Replay protection
ioctl

- Configuration interface
- Using nvlists for extensibility
- Linux uses netlink
  - We have netlink too
- Now
  - Netlink landed after this work
Routing

• Tunnels are not a broadcast domain

• Need to work out which peer to send to

• Second routing lookup based on destination IP of the tunneled packet

• `ovpn_route_peer()`

• Special case

• Only one client
Key rotation

• Handled (mostly) by userspace

• Negotiation (userspace)

• Install new key with OVPN_NEW_KEY cmd

• Switch over with OVPN_SWAP_KEYS

• Each packet contains a key id

• No traffic disruption

• Remove old key with OVPN_DEL_KEY
vnet

• Not a required feature

• But so, so good for testing

• /usr/tests/sys/net/if_ovpn
Performance

- Tested on a Netgate 4100
- Intel® Atom® C3338R with QAT, 2-core @ 1.8 GHz

<table>
<thead>
<tr>
<th>Feature</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>if_tun (AES-NI)</td>
<td>207.3 Mbit/s</td>
</tr>
<tr>
<td>DCO Software</td>
<td>213.1 Mbit/s</td>
</tr>
<tr>
<td>DCO AES-NI</td>
<td>751.2 Mbit/s</td>
</tr>
<tr>
<td>DCO QAT</td>
<td>1,064.8 Mbit/s</td>
</tr>
</tbody>
</table>
Where can I get this?

- OpenVPN 2.6.0
  - Released January 26th 2023
- FreeBSD 14.0
  - Also Linux and Windows
  - "But you don't care"
- pfSense+ 22.05
  - Released 27/06/2022
Thank You
Questions?

Kristof Provost  
kp@FreeBSD.org  
@kp@bsd.network