High-speed traffic encryption on x86_64 with Snabb

FOSDEM 20
Max Rottenkolber
max@mr.gy
@eugeneia_
Max Rottenkolber <max@mr.gy>

Open source hacker, working on Snabb since 2014

Consulting on software networking (in userspace), protocols, optimization...
Vita is a high-performance site-to-site VPN gateway

Fully open source (and hackable!)

 Runs on generic x86_64 server CPUs
Based on **snabb**

Written in a high-level language (Lua)

Made possible by **nlnet Foundation**
while not link.empty(input) do
    local p = link.receive(input)
    if ipv4_ttl(p) > 0 then
        link.transmit(output, p)
    else
        link.transmit(time_exceeded, p)
    end
end
~3 Mpps per core on a modern CPU (duplex)

...or ~5 Gbps of IMIX traffic per core

Medium-term goal: 100 Gbps at 60 byte packets on a generic x86 server
How?

In Snabb-land we like to write software that is both fast and simple

...and we don’t like vendor lock-in

No QuickAssist, crypto cards... Only x86_64!
How?

For crunching numbers (encryption): AES-NI, AVX2 (optimized AES-GCM implementation written in DynASM)

```plaintext
function ghash_mul(Dst, gh, hk, t1, t2, t3)
  | vpcm ulqdq xmm(t1), xmm(gh), xmm(hk), 0x11
  | vpcm ulqdq xmm(t2), xmm(gh), xmm(hk), 0x00
  | vpcm ulqdq xmm(t3), xmm(gh), xmm(hk), 0x01
  | vpcm ulqdq xmm(gh), xmm(gh), xmm(hk), 0x10
  | vpxor xmm(gh), xmm(gh), xmm(t3)
  ...
```
How?

For route lookups (longest prefix match):
Optimized Poptrie implementation (again, DynASM)

```plaintext
function lookup (Dst, Poptrie, keysize)
    if Poptrie.direct_pointing then
        -- v = extract(key, 0, Poptrie.s)
        local direct_mask = bit.lshift(1ULL, Poptrie.s) - 1
        -- v = band(key, direct_mask)
        | mov v_dw, dword [key]
        | and v, direct_mask
    ...
```
How?

RaptorJIT + FFI
(simple and fast implementation of IPsec ESP)

```c
esp_head = ffi.typedef[[
    struct {
        uint32_t spi;
        uint32_t seq_no;
    } __attribute__((packed))
]]
```

```c
esp_tail = ffi.typedef[[
    struct {
        uint8_t pad_length;
        uint8_t next_header;
    } __attribute__((packed))
]]
```
How?

DSL for match-action pipeline, based on pcap-filter(7) language (code generation at runtime)

```clojure
(pf_match.compile(([[match {
ip dst host %s and icmp => icmp4
ip dst host %s => protocol4_unreachable
ip => forward4
arp => arp
otherwise => reject_ethertype
}]}}):format(conf.node_ip4, conf.node_ip4)))
```
How?

Problem: can not parallelize SA

Every packet on an SA gets a unique sequence number

Synchronization problem if spread across cores
How?

Solution: scale out (multiple SAs per route)

RSS on private interface: distribute onto SAs

VMDq on public interface: aggregate SAs
How?

Interface in RSS mode (hash on flows)

private addr 192.168.0.10

Q1

public addr 203.137.0.1

Q2

203.137.0.2

Interface in VMDq mode (virtualized)
Network Drivers

The Snabb way: simple drivers written in Lua

New: Snabb drivers for XDP, Intel AVF
XDP?

Immediate goal: make Vita easily deployable in cloud

XDP:  - a bit heavy to setup, currently some limitations
     - working with kernel upstream looks promising (kudos to Björn Töpel)
AKE (authenticated key exchange)

Cycle SAs often and without loosing packets (perfect forward secrecy)

Low-throughput, but largest attack surface
AKE (authenticated key exchange)

Simple PSK based protocol
(based on the Noise protocol framework)

Using minimal set of cryptographic primitives, constructions and protocol implemented in Lua
AKE (authenticated key exchange)

Alternatively: IKEv2 via StrongSwan

SWITCH engineer Alexander Gall developed StrongSwan plugin+interop with Snabb
Configuration & operation

Based on a YANG model

...includes runtime statistics

module vita-esp-gateway {
  ...
}
Configuration & operation

Query/update configuration via RPC

Query runtime statistics via RPC

$ snabb config get-state vita /gateway-state/private-interface
Thanks!

Get involved:
github.com/snabbco/snabb
github.com/inters/vita

Get support and consulting:
https://inters.co

Email me:
max@mr.gy

Gritty details on my blog:
https://mr.gy/blog