

# xfrm: Is the flow cache still needed?

Florian Westphal  
4096R/AD5FF600 fw@strlen.de  
80A9 20C5 B203 E069 F586  
AE9F 7091 A8D9 AD5F F600

Red Hat

netdev 1.2, Tokyo, October 2016

# Intro

- ▶ kernel has a 'flow cache' to store xfrm bundles
- ▶ xfrm bundle contains everything needed to do ipsec
  - ▶ route info (including e.g. mtu)
  - ▶ cipher info
- ▶ flow cache, like any cache, has its pros and cons
  - ▶ cache hit: can (re)use cached xfrm dst
  - ▶ cache (constantly) overloaded: slower than without cache

## ipsec tput with 4 udp flows per cpu, in Mbit/s, veth

64 byte packets:

cpus	net-next	no-cache		net-next	no-cache	
1	241	198	-18%	1160	1004	-13%
2	469	320	-32%	2270	2002	-12%
3	690	503	-28%	3250	2920	-10%
4	880	688	-22%	4330	4000	-8%

1400 byte packets:

cpus	net-next	no-cache		net-next	no-cache	
1	241	198	-18%	1160	1004	-13%
2	469	320	-32%	2270	2002	-12%
3	690	503	-28%	3250	2920	-10%
4	880	688	-22%	4330	4000	-8%

## ipsec tput with 16k flows per cpu, in MBit/s, veth

64 byte packets:

cpus	net-next	no flowcache	
1	112	190	+70%
2	193	307	+59%
3	305	488	+60%
4	435	668	+53%

1400 byte packets:

cpus	net-next	no flowcache	
4	2700	3920	+45%

Flow cache, 64 byte packets, 4 flows:

Overhead	Shared	Object	Symbol
16.16%	[kernel]	[k]	sha_transform
5.15%	[kernel]	[k]	fib_table_lookup
4.67%	[kernel]	[k]	_aesni_enc1
2.75%	[kernel]	[k]	ip_route_input_noref
2.56%	[dummy]	[k]	dummy_napi_rx
2.50%	[kernel]	[k]	__netif_receive_skb_core
2.49%	[kernel]	[k]	__memset
2.38%	[esp4]	[k]	esp_output
2.24%	[kernel]	[k]	__memcpy
1.90%	[kernel]	[k]	ip_rcv
1.90%	[kernel]	[k]	dst_release
1.77%	[kernel]	[k]	xfrm_output_resume
1.45%	[kernel]	[k]	scatterwalk_copychunks
1.41%	[kernel]	[k]	ip_idents_reserve
1.27%	[kernel]	[k]	__xfrm_route_forward
1.26%	[kernel]	[k]	flow_cache_lookup
1.16%	[kernel]	[k]	_decode_session4

No flow cache, 64 byte packets:

Overhead	Shared	Object	Symbol
11.92%	[kernel]	[k]	sha_transform
5.66%	[kernel]	[k]	fib_table_lookup
3.16%	[kernel]	[k]	_aesni_enc1
2.52%	[kernel]	[k]	xfrm_resolve_and_create_bundle
2.29%	[kernel]	[k]	xfrm4_dst_destroy
2.02%	[kernel]	[k]	__netif_receive_skb_core
2.01%	[kernel]	[k]	dst_release
2.00%	[kernel]	[k]	xfrm_state_find
1.98%	[kernel]	[k]	ip_route_input_noref
1.92%	[kernel]	[k]	__ip_route_output_key_hash
1.89%	[kernel]	[k]	xfrm_policy_lookup_bytype
1.81%	[dummy]	[k]	dummy_napi_rx
1.78%	[kernel]	[k]	__memcpy
1.76%	[esp4]	[k]	esp_output
1.68%	[kernel]	[k]	__memset
1.53%	[kernel]	[k]	ip_rcv
1.29%	[kernel]	[k]	dst_destroy

# ipsec tput with 4 udp flows per cpu, in Mbit/s, benet

64 byte packets:

cpus	net-next	no-cache	
1	226	195	-14%
2	431	341	-21%
3	591	510	-14%
4	790	620	-21%

1500 byte packets:

net-next	no-cache	
1116	1007	-10%
2100	2004	-4%
3210	2910	-8%
4330	3920	-10%

Flow cache, 64 byte packets, 4 flows per core

Overhead	Shared Object	Symbol
15.73%	[kernel]	[k] sha_transform
4.45%	[kernel]	[k] _aesni_enc1
3.39%	[kernel]	[k] fib_table_lookup
2.78%	[dummy]	[k] dummy_napi_rx
2.69%	[kernel]	[k] _raw_spin_lock
2.61%	[kernel]	[k] __memset
2.49%	[esp4]	[k] esp_output
2.30%	[kernel]	[k] __memcpy
2.02%	[kernel]	[k] ip_route_input_noref
[..]		
1.85%	[kernel]	[k] dst_release
1.69%	[be2net]	[k] be_xmit_enqueue
[..]		
1.32%	[kernel]	[k] __xfrm_route_forward
1.32%	[kernel]	[k] _decode_session4
1.22%	[kernel]	[k] flow_cache_lookup

Without Flow cache, 64 byte packets, 4 flows per core

Overhead	Shared	Object	Symbol
12.80%	[kernel]	[k]	sha_transform
4.04%	[kernel]	[k]	fib_table_lookup
2.68%	[kernel]	[k]	_aesni_enc1
2.19%	[kernel]	[k]	xfrm_resolve_and_create_bundle
1.84%	[kernel]	[k]	xfrm_state_find
1.79%	[kernel]	[k]	__memcpy
1.79%	[esp4]	[k]	esp_output
1.76%	[dummy]	[k]	dummy_napi_rx
1.72%	[kernel]	[k]	xfrm4_dst_destroy
1.68%	[kernel]	[k]	xfrm_policy_lookup_bytype
1.59%	[kernel]	[k]	dst_release
1.56%	[kernel]	[k]	ip_route_input_noref
1.52%	[be2net]	[k]	be_xmit_enqueue
1.41%	[kernel]	[k]	__memset
1.40%	[kernel]	[k]	__netif_receive_skb_core
1.32%	[kernel]	[k]	_raw_spin_lock
1.22%	[kernel]	[k]	dst_destroy

- ▶ Several functions will be called per-packet rather than when flow is added to cache
  - ▶ `xfrm4_get_saddr` (creates temporary rt dst)
  - ▶ MTU init (`xfrm_state_mtu`, grabs `xfrm_state->lock`)<sup>1</sup>
- ▶ most of the perf hit seems to come from dst creation/destruction of xfrm dsts (and not from e.g. overly expensive spi lookup)

---

<sup>1</sup>lock doesn't seem to be needed

Script used on transmit side (for rx swap ME/LOCAL/REMOTE)

```
ip addr add 192.168.13.1/24 dev dummy0
```

```
ip addr add dev ens4f0 192.168.11.2/24
```

```
ME=192.168.11.2
```

```
REMOTE=192.168.11.1
```

```
NETLOCAL=192.168.13.0/24
```

```
NETREMOTE=192.168.12.0/24
```

```
KEY=0x1c0cf2bb72ce67ba245b593076dfb27a
```

```
ip xfrm state add src $ME dst $REMOTE spi 1 proto esp \  
mode tunnel reqid 1 auth sha1 $KEY enc 'cbc(aes)' "$KEY"
```

```
ip xfrm state add dst $ME src $REMOTE spi 1 proto esp \  
mode tunnel reqid 1 auth sha1 $KEY enc 'cbc(aes)' "$KEY"
```

```
ip xfrm policy add src $NETLOCAL dst $NETREMOTE dir out \  
tmpl dst $REMOTE proto esp mode tunnel reqid 1
```

```
ip xfrm policy add src $NETREMOTE dst $NETLOCAL dir in \  
tmpl dst $ME proto esp mode tunnel reqid 1
```