

# Linux IPSEC Scaling: Seattle Edition



LCA 2007

Sydney, Australia

David S. Miller

([davem@davemloft.net](mailto:davem@davemloft.net))

# Overview

- General Architecture
- Core correctness issues
- Jamal's test case
- Initial implementation vs. usage analysis tuning
- Fixing the obvious problems
- Usage analysis driven lookup implementation

# Top-level Architecture

- Policy engine sits at root, lookups via flow cache
- Policy refers to “state” and “template” pools
- Templates gather states to create IPSEC routes
- States are specific IPSEC encaps/decaps
- Thus, many indirect references between the various object pool spaces.

# Core Correctness Issues

- State deletion:
  - Must ensure that any cache routes built from this state are no longer used.
- State addition:
  - Rule shadowing. New state can be a better match for flows than existing states.
- Core issue: active vs. passive flushing

# Jamal's test case

- Stress test of configuration changes
- Several modes of operation
  - One policy, many states
  - Many policies
  - Many policies, several states per-policy
- In short, we sucked...
- Super-DaveM to the rescue!

# Usage Analysis

- Jamal's results were pleasing, why?
- Keep it simple from the beginning.
- If you don't know how it will be used, don't optimize a thing!
- Thusly, with our IPSEC implementation we started with dumb algorithms.
- Now we know, and can optimize properly.

# Flow Identities

- Addresses
- Ports
- Protocols
- Prefixes
- No general solution
- 2-way prefixed keys are doable
- N-way is not
- Therefore, real usage is critical to optimize these tables.

# Most obvious problems, refcnts

- xfrm\_state objects were overly refcounted
- All that matters is the route reference.
- Everything else is superfluous.
  - Insert: hash add, start timers, etc.
  - Delete: hash remove, wait refcount==0
  - Kill: del\_timer\_sync(), kfree()
- Kills 8 atomics, need for full IPSEC route flush

# ESP Bogosity

- `get_random_bytes()` in the profiles
- WTF?
- Crypto IV initialization for every state
- Only needed for output, and only on first output packet.
- Waste of cpu cycles and random pool entropy
- Fix: Defer to first output packet

# xfrm\_state hash tables

- Dynamic table growth a must.
- Two input hashes:
  - SPI + daddr + protocol
  - SADDR + family
- Output hash: DADDR + SADDR, non-prefixed
  - Used to build output routes from templates.

# xfrm\_policy tables

- Non-trivial problem
- Existing algorithm really dumb, single linked list
- Prefixing is arbitrary
- Exploit common usage:
  - Small number of prefixed “network” policies
  - Arbitrary number of non-prefixed “precise” entries

# Policy Table Implementation

- Idea due to Alexey Kuznetsov
- Keep linked list for “imprecise” prefixed entries
- Create hash table for non-prefixed entries
- Bad Algorithm: (precise || imprecise)
- Good Algorithm: Allow matching imprecise entries to trump precise ones based upon priority if necessary.

# Cheap State Coherency

- Used to flush everything on delete.
- Instead, on delete, do nothing, allow natural garbage collection to purge entries.
- Used to flush everything on add.
- Instead, use generation counts:
  - xfrm\_state and xfrm\_dst generation ID, add bumps
  - state/dst generation ID mis-match invalidates

# Packet Path Issues

- Policy flow cache is sub-optimal:
  - Lookup gives policy, not a route
  - Socket policies are “weird” and not hashed
  - Sub-policies not in the cache
- Pre-computed cached routes (aka. “bundles”) sit on linked list, maybe hash?

# IPSEC Output Path

- Normal IPV4/IPV6 route resolution
- IPSEC policy lookup:
  - Check for socket policy, if any
  - Interrogate flow cache
  - If policy action is BLOCK, stop
  - Find bundle, else build one using templates
  - No matching template? Ask key manager.

# It's Too Slow

- Up to 5 lookups per-packet
- Stuffing cached routes in the flow cache entries can get us down to 2
- But we can do better
- Look into system-wide grand unified flow cache schemes
- Will help implement things like netchannels

# Other Networking Topics

- TCP socket too big
- tcp\_ack() is expensive, queue aggregation
- Trie-in-routing-cache, Aka “Trash”
- MSI-X interrupt targetting
- GUFEC and NetChannels
- RCU and dynamic sizing TCP hashes

# Summary

- Premature optimization == evil
- Do not optimize while blindfolded
- Use what you know
- Do not reference count like an accountant
- The best coherency scheme may be none at all

# Thank You

- Linux Conference Australia
- The whole Linux netdev team:
  - Rusty, Thomas, Herbert, Jamal, Yoshifuji, Alexey, James M., Robert, Patrick, Harald, Arnaldo, etc.
- Linus, of course
- Nintendo Wii
- Mad props to UNSW Slarken Society, oath!