Perl iptables interface
CPAN IPTables::libiptc

Netfilter Workshop 2011
d.24/8-2011

by
Jesper Dangaard Brouer <jdb@comx.dk>
Master of Computer Science
ComX Networks A/S

Who am I

- Name: Jesper Dangaard Brouer
  - Edu: Computer Science for Uni. Copenhagen
    - Focus on Network, Dist. sys and OS
  - Linux user since 1996, professional since 1998
    - Sysadm, Kernel Developer, Embedded
  - OpenSource projects, author of
      - ADSL-optimizer
      - CPAN IPTables::libiptc
      - IPTV-Analyzer
    - Patches accepted into
      - Linux kernel, iproute2, iptables, libpcap and Wireshark

Why name: IPTables::libiptc

- The libiptc comes from the **ipt**ables **c**ache library
  - Perl module started linking with this C-library
  - Now (> v0.51) dynamic loading of
    - libiptc.so
    - libxtables.so
  - Still compile and include iptables.c
    - In several versions
    - Just to call do_command()
  - No need to support extensions in Perl
    - Dyn-loaded by iptables.c

Why is it so fast?

- Take advantage of how iptables talk to kernel
  - Iptables/libiptc transfers the **entire** ruleset
    - (init) from kernel to userspace
    - make several ruleset changes
    - (commit) back from userspace to kernel
      - Atomic commit in kernel
  - Iptables command approach is stupid
    - It only perform **one** change
      - Init → one-change → commit
    - init+commit expensive operations

How fast: stats(1)

- Completely rebuilding access control rules one machine
    - Number of calls: 236645
    - Libiptc time used : 3.95667362 sec
    - Average per call: 0.00001671 sec
    - Perl total time: 16.5 sec
- Ruleset size (*only* filter table):
    - (Note: Both Access + Customer-firewall(bloated))
    - Chains: 19886
    - Rules: 79127
    - Memory 19120128 =~ 19 MB
        - Times 16 CPUs = 304 MB

Detailed stats(2)

- Statistics per command action:

Command	**Calls**	**time-total**	**time-average**
init	1	0.41089988s	0.41089988s
commit	1	0.22868109s	0.22868109s
set_policy	1	0.00001788s	0.00001788s
list_rules_IPs	4015	0.04738951s	0.00001180s
flush_entries	4025	0.01621175s	0.00000403s
insert_rule	15863	0.65369678s	0.00004121s
append_rule	15907	0.33825088s	0.00002126s
delete_rule	31723	1.61556935s	0.00005093s
is_chain	165109	0.64595652s	0.00000391s

- Old method: Command line iptables 236645 calls,
  - init+commit overhead: 0.639 sec * 236645 = 42 hours!

What could I wish for

- Some shared lib exporting iptables do_command()
- Not too many API/ABI changes, please.
- Locking around init() and commit()
  - Concurrent iptables call can clash
    - Mostly one will fail during commit()
    - Risk the wrong ruleset gets comitted.

Future development

- Wrapper module: IPTables::Interface
  - Handles locking + singleton object init
  - Code in my SubnetSkeleton code, see:
  - https://github.com/netoptimizer/IPTables-SubnetSkeleton
  - Move module to CPAN
    - Perhaps in IPTables::libiptc package?
- Keep up with iptables versions
- Better exit on error handling

Status

- IPTables::libiptc version 0.51
  - supporting iptables *above* version 1.4.3.2
    - Due to API changes by Jan Engelhardt
  - up-to iptables version 1.4.10
    - Due to new API changes

The End

- Is anybody using Perl and iptables?
  - Using my module?
    - Please inform me: jdb@comx.dk
- CPAN link:
  - http://search.cpan.org/perldoc?IPTables::libiptc
- Code Git tree:
  - https://github.com/netoptimizer/CPAN-IPTables-libiptc