sysctl

sysctl is a tool for examining and changing kernel parameters at runtime (package procps-ng (https://archlinux.org/packages/?name=procps-ng) in official repositories). sysctl is implemented in procfs, the virtual process file system at `/proc/` .

Contents

Configuration

Note: From version 207 and 21x, systemd only applies settings from `/etc/sysctl.d/*.conf` and `/usr/lib/sysctl.d/*.conf` . If you had customized `/etc/sysctl.conf` , you need to rename it as `/etc/sysctl.d/99-sysctl.conf` . If you had e.g. `/etc/sysctl.d/foo` , you need to rename it to `/etc/sysctl.d/foo.conf` .

The **sysctl** preload/configuration file can be created at `/etc/sysctl.d/99-sysctl.conf`. For systemd, `/etc/sysctl.d/` and `/usr/lib/sysctl.d/` are drop-in directories for kernel sysctl parameters. The naming and source directory decide the order of processing, which is important since the last parameter processed may override earlier ones. For example, parameters in a `/usr/lib/sysctl.d/50-default.conf` will be overriden by equal parameters in `/etc/sysctl.d/50-default.conf` and any configuration file processed later from both directories.

To load all configuration files manually, execute:

```
# sysctl --system
```

which will also output the applied hierarchy. A single parameter file can also be loaded explicitly with:

```
# sysctl --load=filename.conf
```

See the new configuration files (http://0pointer.de/blog/projects/the-new-configuration-files) and more specifically `sysctl.d(5)` (https://man.archlinux.org/man/sysctl.d.5) for more information.

The parameters available are those listed under `/proc/sys/`. For example, the `kernel.sysrq` parameter refers to the file `/proc/sys/kernel/sysrq` on the file system. The `sysctl --all` command can be used to display all currently available values.

Note: If you have the kernel documentation installed (`linux-docs` (https://archlinux.org/packages/?name=linux-docs)), you can find detailed information about sysctl settings in `/usr/lib/modules/$(uname -r)/build/Documentation/admin-guide/sysctl/`. An online version is referred by the #See also section of this article. It is highly recommended reading these before changing sysctl settings.

Settings can be changed through file manipulation or using the `sysctl` utility. For example, to temporarily enable the magic SysRq key:

```
# sysctl kernel.sysrq=1
```

or:

```
# echo "1" > /proc/sys/kernel/sysrq
```

See Linux kernel documentation (https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html) for details about `kernel.sysrq`.

To preserve changes between reboots, add or modify the appropriate lines in `/etc/sysctl.d/99-sysctl.conf` or another applicable parameter file in `/etc/sysctl.d/`.

Tip: Some parameters that can be applied may depend on kernel modules which in turn might not be loaded. For example parameters in `/proc/sys/net/bridge/*` depend on the `br_netfilter` module. If it is not loaded at runtime (or after a reboot), those will *silently* not be applied. See Kernel modules.

Security

See also Security#Kernel hardening, as well as the rest of this article.

Networking

Improving performance

Increasing the size of the receive queue.

The received frames will be stored in this queue after taking them from the ring buffer on the network card.

Increasing this value for high speed cards may help prevent losing packets:

```
net.core.netdev_max_backlog = 16384
```

Note: In real time application like SIP routers, this option requires a high speed CPU otherwise the data in the queue will be out of date.

Increase the maximum connections

The upper limit on how many connections the kernel will accept (default 128):

```
net.core.somaxconn = 8192
```

Note: Kernel 5.4 raised the default value to 4096.[1] (https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/commit/?id=19f92a030ca6d772ab44b22ee6a01378a8cb32d4)

Warning: Increasing this value may only increase performance on high-loaded servers and may cause as slow processing rate (e.g. a single threaded blocking server) or insufficient number of worker threads/processes [2] (https://serverfault.com/questions/518862/will-increasing-net-core-somaxconn-make-a-difference/519152).

Increase the memory dedicated to the network interfaces

The default the Linux network stack is not configured for high speed large file transfer across WAN links (i.e. handle more network packets) and setting the correct values may save memory resources:

```
net.core.rmem_default = 1048576
net.core.rmem_max = 16777216
net.core.wmem_default = 1048576
net.core.wmem_max = 16777216
net.core.optmem_max = 65536
net.ipv4.tcp_rmem = 4096 1048576 2097152
net.ipv4.tcp_wmem = 4096 65536 16777216
```

It is also possible increase the default `4096` UDP limits:

```
net.ipv4.udp_rmem_min = 8192
net.ipv4.udp_wmem_min = 8192
```

See the following sources for more information and recommend values:

- http://www.nateware.com/linux-network-tuning-for-2013.html
- https://blog.cloudflare.com/the-story-of-one-latency-spike/

Enable TCP Fast Open

TCP Fast Open is an extension to the transmission control protocol (TCP) that helps reduce network latency by enabling data to be exchanged during the sender's initial TCP SYN [3] (https://www.keycdn.com/support/tcp-fast-open/). Using the value `3` instead of the default `1` allows TCP Fast Open for both incoming and outgoing connections:

```
net.ipv4.tcp_fastopen = 3
```

Tweak the pending connection handling

`tcp_max_syn_backlog` is the maximum queue length of pending connections 'Waiting Acknowledgment'.

In the event of a synflood DOS attack, this queue can fill up pretty quickly, at which point **TCP SYN cookies** will kick in allowing your system to continue to respond to legitimate traffic, and allowing you to gain access to block malicious IPs.

If the server suffers from overloads at peak times, you may want to increase this value a little bit:

```
net.ipv4.tcp_max_syn_backlog = 8192
```

`tcp_max_tw_buckets` is the maximum number of sockets in TIME_WAIT state.

After reaching this number the system will start destroying the socket that are in this state.

Increase this to prevent simple DOS attacks:

```
net.ipv4.tcp_max_tw_buckets = 2000000
```

`tcp_tw_reuse` sets whether TCP should reuse an existing connection in the TIME-WAIT state for a new outgoing connection if the new timestamp is strictly bigger than the most recent timestamp recorded for the previous connection.

This helps avoid from running out of available network sockets:

```
net.ipv4.tcp_tw_reuse = 1
```

Specify how many seconds to wait for a final FIN packet before the socket is forcibly closed. This is strictly a violation of the TCP specification, but required to prevent denial-of-service attacks. In Linux 2.2, the default value was 180 **[4] (https://access.redhat.com/solutions/41776)**:

```
net.ipv4.tcp_fin_timeout = 10
```

`tcp_slow_start_after_idle` sets whether TCP should start at the default window size only for new connections or also for existing connections that have been idle for too long.

This setting kills persistent single connection performance and could be turned off:

```
net.ipv4.tcp_slow_start_after_idle = 0
```

Change TCP keepalive parameters

TCP keepalive is a mechanism for TCP connections that help to determine whether the other end has stopped responding or not. TCP will send the keepalive probe that contains null data to the network peer several times after a period of idle time. If the peer does not respond, the socket will be closed automatically. By default, TCP keepalive process waits for two hours (7200 secs) for socket activity before sending the first keepalive probe, and then resend it every 75 seconds. As long as there is TCP/IP socket communications going on and active, no keepalive packets are needed.

Note: With the following settings, your application will detect dead TCP connections after 120 seconds (60s + 10s + 10s + 10s + 10s + 10s + 10s).

```
net.ipv4.tcp_keepalive_time = 60
net.ipv4.tcp_keepalive_intvl = 10
net.ipv4.tcp_keepalive_probes = 6
```

Enable MTU probing

The longer the **maximum transmission unit (MTU)** the better for performance, but the worse for reliability.

This is because a lost packet means more data to be retransmitted and because many routers on the Internet cannot deliver very long packets:

```
net.ipv4.tcp_mtu_probing = 1
```

See **https://blog.cloudflare.com/path-mtu-discovery-in-practice/** for more information.

TCP timestamps

Warning: TCP timestamps protect against wrapping sequence numbers (at gigabit speeds) and round trip time calculation implemented in TCP. It is not recommended to turn off TCP timestamps as it may cause a security risk **[5] (https://access.redhat.com/sites/default/files/attachments/20150325_network_performance_tuning.pdf)**.

Disabling timestamp generation will reduce spikes and may give a performance boost on gigabit networks:

```
net.ipv4.tcp_timestamps = 0
```

Enable BBR

The **BBR congestion control algorithm** can help achieve higher bandwidths and lower latencies for internet traffic. First, **load** the `tcp_bbr` module.

Note: **BBR GitHub says (https://github.com/google/bbr/blob/master/README)**, "This is not an official Google product."

```
net.core.default_qdisc = cake
net.ipv4.tcp_congestion_control = bbr
```

Increase the Ephemeral port range

The **Wikipedia:Ephemeral port** is typically used by the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), or the Stream Control Transmission Protocol (SCTP) as the port assignment for the client end of a client–server communication.

```
net.ipv4.ip_local_port_range = 30000 65535
```

TCP/IP stack hardening

The following specifies a parameter set to tighten network security options of the kernel for the IPv4 protocol and related IPv6 parameters where an equivalent exists.

For some use-cases, for example using the system as a **router**, other parameters may be useful or required as well.

TCP SYN cookie protection

Helps protect against SYN flood attacks. Only kicks in when `net.ipv4.tcp_max_syn_backlog` is reached. More details at, for example, [6] (https://cateee.net/lkddb/web-lkddb/SYN_COOKIES.html). As of `linux (https://archlinux.org/packages/?name=linux)` 5.10, it is set by default.

```
net.ipv4.tcp_syncookies = 1
```

TCP rfc1337

Protect against tcp time-wait assassination hazards, drop RST packets for sockets in the time-wait state. Not widely supported outside of Linux, but conforms to RFC:

```
net.ipv4.tcp_rfc1337 = 1
```

Reverse path filtering

By enabling reverse path filtering, the kernel will do source validation of the packets received from all the interfaces on the machine. This can protect from attackers that are using IP spoofing methods to do harm.

The kernel's default value is `0` (no source validation), but systemd ships `/usr/lib/sysctl.d/50-default.conf` that sets `net.ipv4.conf.all.rp_filter` to `2` (loose mode)[8] (https://github.com/systemd/systemd/pull/10971).

The following will set the reverse path filtering mechanism to value `1` (strict mode):

```
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
```

The relationship and behavior of `net.ipv4.conf.default.*`, `net.ipv4.conf.`*`interface`*`.*` and `net.ipv4.conf.all.*` is explained in ip-sysctl.html (https://www.kernel.org/doc/html/latest/networking/ip-sysctl.html).

Log martian packets

A martian packet is an IP packet which specifies a source or destination address that is reserved for special-use by Internet Assigned Numbers Authority (IANA). See Reserved IP addresses for more details.

Often martian and unroutable packet may be used for a dangerous purpose. Logging these packets for further inspection may be useful [9] (https://www.cyberciti.biz/faq/linux-log-suspicious-martian-packets-un-routable-source-addresses/):

```
net.ipv4.conf.default.log_martians = 1
net.ipv4.conf.all.log_martians = 1
```

Note: This can fill up your logs with a lot of information, it is advisable to only enable this for testing.

Disable ICMP redirects

Background is at What are ICMP redirects? Should they be blocked? (https://askubuntu.com/questions/118273/what-are-icmp-redirects-and-should-they-be-blocked)

To disable ICMP redirect acceptance:

```
net.ipv4.conf.all.accept_redirects = 0
```

```
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
```

To disable ICMP redirect sending when on a non router:

```
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
```

Ignore ICMP echo requests

To disable ICMP echo (aka ping) requests:

```
net.ipv4.icmp_echo_ignore_all = 1
net.ipv6.icmp.echo_ignore_all = 1
```

Note: Beware this may cause issues with monitoring tools and/or applications relying on ICMP echo responses.

Other

Allow unprivileged users to create IPPROTO_ICMP sockets

The IPPROTO_ICMP (**icmp(7) (https://man.archlinux.org/man/icmp.7)**) socket type adds the possibility to send ICMP_ECHO messages and receive corresponding ICMP_ECHOREPLY messages without the need to open a **raw(7) (https://man.archlinux.org/man/raw.7)** socket, an operation which requires the CAP_NET_RAW **capability** or the SUID bit with a proper privileged owner. These ICMP_ECHO messages are sent by the **ping** application thus making the IPPROTO_ICMP socket also known as ping socket in addition to ICMP Echo socket.

`ping_group_range` determines the GID range of groups which their users are allowed to create IPPROTO_ICMP sockets. Additionally, the owner of the CAP_NET_RAW capability is also allowed to create IPPROTO_ICMP sockets.
By default this range is `1 0` which means no one is allowed to create IPPROTO_ICMP sockets except root.
To take advantage of this setting programs which currently uses raw sockets need to ported to use IPPROTO_ICMP sockets instead.
For example, QEMU uses IPPROTO_ICMP for SLIRP aka User-mode networking, so allowing the user running QEMU to create IPPROTO_ICMP sockets means it's possible to ping from the guest.

To allow only users which are members of the group with GID 100 to create IPPROTO_ICMP sockets:

```
net.ipv4.ping_group_range = 100 100
```

To allow all the users in the system to create IPPROTO_ICMP sockets:

```
net.ipv4.ping_group_range = 0 65535
```

Virtual memory

There are several key parameters to tune the operation of the **virtual memory** subsystem of the Linux kernel and the write out of dirty data to disk. See the official **Linux kernel documentation (https://www.kernel.org/doc/html/latest/admin-guide/sysctl/vm.html)** for more information. For example:

- `vm.dirty_ratio = 10`

  Contains, as a percentage of total available memory that contains free pages and reclaimable pages, the number of pages at which a process which is generating disk writes will itself start writing out dirty data.

- `vm.dirty_background_ratio = 5`

  Contains, as a percentage of total available memory that contains free pages and reclaimable pages, the number of pages at which the background kernel flusher threads will start writing out dirty data.

As noted in the comments for the parameters, one needs to consider the total amount of RAM when setting these values. For example, simplifying by taking the installed system RAM instead of available memory:

Warning:

- Higher ratio values may increase performance, it also increases the risk of data loss.
- Setting this value to `0` may cause higher latency on disks and spikes.

See **https://lonesysadmin.net/2013/12/22/better-linux-disk-caching-performance-vm-dirty_ratio/** for more information.

- Consensus is that setting `vm.dirty_ratio` to 10% of RAM is a sane value if RAM is say 1 GB (so 10% is 100 MB). But if the machine has much more RAM, say 16 GB (10% is 1.6 GB), the percentage may be out of proportion as it becomes several seconds of writeback on spinning disks. A more sane value in this case may be `3` (3% of 16 GB is approximately 491 MB).
- Similarly, setting `vm.dirty_background_ratio` to `5` may be just fine for small memory values, but again, consider and adjust accordingly for the amount of RAM on a particular system.

VFS cache

Decreasing the **virtual file system (https://www.kernel.org/doc/html/latest/filesystems/vfs.html)** (VFS) cache parameter value may improve system responsiveness:

- `vm.vfs_cache_pressure = 50`

  The value controls the tendency of the kernel to reclaim the memory which is used for caching of directory and inode objects (VFS cache). Lowering it from the default value of 100 makes the kernel less inclined to reclaim VFS cache (do not set it to 0, this may produce out-of-memory conditions).

MDADM

See **RAID#Change sync speed limits**.

Troubleshooting

Small periodic system freezes

Set dirty bytes to small enough value (for example 4M):

```
vm.dirty_background_bytes = 4194304
vm.dirty_bytes = 4194304
```

Note: The `dirty_background_bytes` and `dirty_bytes` parameters are counterparts of `dirty_background_ratio` and `dirty_ratio` (as seen in **#Virtual memory**). Only one of the parameters may be specified at a time.

See also

- **`sysctl(8)` (https://man.archlinux.org/man/sysctl.8)** and **`sysctl.conf(5)` (https://man.archlinux.org/man/sysctl.conf.5)**
- **Linux kernel documentation for /proc/sys/ (https://www.kernel.org/doc/html/latest/admin-guide/sysctl/index.html)**
- Kernel Documentation: **IP Sysctl (https://www.kernel.org/doc/html/latest/networking/ip-sysctl.html)**
- **Kernel network parameters for sysctl (https://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.kernel.html)**
- **sysctl-explorer.net – an initiative to facilitate the access of Linux' sysctl reference documentation (https://sysctl-explorer.net)**
- **Disable Source Routing - Red Hat Customer Portal (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-server_security-disable-source-routing)**
- **SUSE handbook about Security Features in the Kernel (https://www.suse.com/documentation/sles-12/book_hardening/data/sec_sec_prot_general_kernel.html)**