Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'slirp4netns.1' command

$ man slirp4netns.1

SLIRP4NETNS(1) User Commands SLIRP4NETNS(1)

NAME

slirp4netns - User-mode networking for unprivileged network namespaces

SYNOPSIS

slirp4netns [OPTION]... PID|PATH [TAPNAME]

DESCRIPTION

slirp4netns provides user-mode networking ("slirp") for network name?

spaces.

Unlike veth(4), slirp4netns does not require the root privileges on the

host.

Default configuration:

? MTU: 1500

? CIDR: 10.0.2.0/24

? Gateway/Host: 10.0.2.2 (network address + 2)

? DNS: 10.0.2.3 (network address + 3)

? DHCP begin: 10.0.2.15 (network address + 15)

? DHCP end: 10.0.2.30 (network address + 30)

? IPv6 CIDR: fd00::/64

? IPv6 Gateway/Host: fd00::2

? IPv6 DNS: fd00::3

OPTIONS

-c, --configure bring up the TAP interface. IP will be set to

10.0.2.100 (network address + 100) by default. IPv6 will be set to a

random address. Starting with v0.4.0, the loopback interface (lo) is brought up as well.

-e, --exit-fd=FD specify the FD for terminating slirp4netns. When the FD is specified, slirp4netns exits when a poll(2) event happens on the FD.

-r, --ready-fd=FD specify the FD to write to when the initialization steps are finished. When the FD is specified, slirp4netns writes "1" to the FD and close the FD. Prior to v0.4.0, the FD was written after the network configuration (-c) but before the API socket configuration (-a).

-m, --mtu=MTU (since v0.2.0) specify MTU (max=65521).

-6, --enable-ipv6 (since v0.2.0, EXPERIMENTAL) enable IPv6

-a, --api-socket (since v0.3.0) API socket path

--cidr (since v0.3.0) specify CIDR, e.g. 10.0.2.0/24

--disable-host-loopback (since v0.3.0) prohibit connecting to 127.0.0.1:* on the host namespace

--netns-type=TYPE (since v0.4.0) specify network namespace type ([path|pid], default=pid)

--userns-path=PATH (since v0.4.0) specify user namespace path

--enable-sandbox (since v0.4.0) enter the user namespace and create a new mount namespace where only /etc and /run are mounted from the host. Requires /etc/resolv.conf not to be a symlink to a file outside /etc and /run.

When running as the root, the process does not enter the user namespace but all the capabilities except CAP_NET_BIND_SERVICE are dropped.

--enable-seccomp (since v0.4.0, EXPERIMENTAL) enable seccomp(2) to limit syscalls. Typically used in conjunction with --enable-sandbox.

--outbound-addr=IPv4 (since v1.1.0, EXPERIMENTAL) specify outbound ipv4 address slirp should bind to

--outbound-addr=INTERFACE (since v1.1.0, EXPERIMENTAL) specify outbound interface slirp should bind to (ipv4 traffic only)

--outbound-addr=IPv6 (since v1.1.0, EXPERIMENTAL) specify outbound ipv6 address slirp should bind to

--outbound-addr6=INTERFACE (since v1.1.0, EXPERIMENTAL) specify out?

bound interface slirp should bind to (ipv6 traffic only)

--disable-dns (since v1.1.0) disable built-in DNS (10.0.2.3 by default)

--macaddress (since v1.1.9) specify MAC address of the TAP interface

(only valid with -c)

--target-type=TYPE (since v1.2.0) specify the target type

([netns|bess], default=netns).

The bess mode (since v1.2.0, EXPERIMENTAL) is expected to be used with

User Mode Linux. The bess mode conflicts with --configure, --netns-

type, and --userns-path.

-h, --help (since v0.2.0) show help and exit

-v, --version (since v0.2.0) show version and exit

EXAMPLE

Terminal 1: Create user/network/mount namespaces

(host)$ unshare --user --map-root-user --net --mount

(namespace)$ echo $$ > /tmp/pid

In this documentation, we use (host)$ as the prompt of the host shell,

(namespace)$ as the prompt of the shell running in the namespaces.

If unshare fails, try the following commands (known to be needed on De?

bian, Arch, and old CentOS 7.X):

(host)$ sudo sh -c 'echo "user.max_user_namespaces=28633" >> /etc/sysctl.d/userns.conf'

(host)$ if [-f /proc/sys/kernel/unprivileged_userns_clone]; then sudo sh -c 'echo "kernel.unprivileged_userns_clone=1" >> /etc/sysctl.d/userns.conf'; fi

(host)$ sudo sysctl --system

Terminal 2: Start slirp4netns

(host)$ slirp4netns --configure --mtu=65520 $(cat /tmp/pid) tap0

starting slirp, MTU=65520

Terminal 1: Make sure tap0 is configured and connected to the Internet

(namespace)$ ip a

1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

3: tap0: <BROADCAST,UP,LOWER_UP> mtu 65520 qdisc fq_codel state UNKNOWN group default qlen 1000

link/ether c2:28:0c:0e:29:06 brd ff:ff:ff:ff:ff:ff

```
    inet 10.0.2.100/24 brd 10.0.2.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::c028:cff:fe0e:2906/64 scope link
       valid_lft forever preferred_lft forever
(namespace)$ echo "nameserver 10.0.2.3" > /tmp/resolv.conf
(namespace)$ mount --bind /tmp/resolv.conf /etc/resolv.conf
(namespace)$ curl https://example.com
```

Bind-mounting /etc/resolv.conf is only needed when /etc/resolv.conf on the host refers to loopback addresses (127.0.0.X, typically dnsmasq(8) or systemd-resolved.service(8)) that cannot be accessed from the name? space.

If your /etc/resolv.conf on the host is managed by networkmanager(8) or systemd-resolved.service(8), you might need to mount a new filesystem on /etc instead, so as to prevent the new /etc/resolv.conf from being unmounted unexpectedly when /etc/resolv.conf on the host is regener? ated.

```
(namespace)$ mkdir /tmp/a /tmp/b
(namespace)$ mount --rbind /etc /tmp/a
(namespace)$ mount --rbind /tmp/b /etc
(namespace)$ mkdir /etc/.ro
(namespace)$ mount --move /tmp/a /etc/.ro
(namespace)$ cd /etc
(namespace)$ for f in .ro/*; do ln -s $f $(basename $f); done
(namespace)$ rm resolv.conf
(namespace)$ echo "nameserver 10.0.2.3" > resolv.conf
(namespace)$ curl https://example.com
```

These steps can be simplified with rootlesskit --copy-up=/etc if root? lesskit is installed:

```
(host)$ rootlesskit --net=slirp4netns --copy-up=/etc bash
(namespace)$ cat /etc/resolv.conf
nameserver 10.0.2.3
```

ROUTING PING PACKETS

To route ping packets, you may need to set up net.ipv4.ping_group_range

properly as the root.

e.g.

(host)$ sudo sh -c 'echo "net.ipv4.ping_group_range=0 2147483647" > /etc/sysctl.d/ping_group_range.conf'

(host)$ sudo sysctl --system

FILTERING CONNECTIONS

By default, ports listening on INADDR_LOOPBACK (127.0.0.1) on the host are accessible from the child namespace via the gateway (default: 10.0.2.2). --disable-host-loopback can be used to prohibit connecting to INADDR_LOOPBACK on the host.

However, a host loopback address might be still accessible via the built-in DNS (default: 10.0.2.3) if /etc/resolv.conf on the host refers to a loopback address. You may want to set up iptables for limiting access to the built-in DNS in such a case.

(host)$ nsenter -t $(cat /tmp/pid) -U --preserve-credentials -n

(namespace)$ iptables -A OUTPUT -d 10.0.2.3 -p udp --dport 53 -j ACCEPT

(namespace)$ iptables -A OUTPUT -d 10.0.2.3 -j DROP

API SOCKET

slirp4netns can provide QMP-like API server over an UNIX socket file:

(host)$ slirp4netns --api-socket /tmp/slirp4netns.sock ...

add_hostfwd: Expose a port (IPv4 only)

(namespace)$ json='{"execute": "add_hostfwd", "arguments": {"proto": "tcp", "host_addr": "0.0.0.0", "host_port": 8080, "guest_addr": "10.0.2.100", "guest_port": 80}}'

(namespace)$ echo -n $json | nc -U /tmp/slirp4netns.sock

{"return": {"id": 42}}

If host_addr is not specified, then it defaults to "0.0.0.0".

If guest_addr is not specified, then it will be set to the default ad?

dress that corresponds to --configure.

list_hostfwd: List exposed ports

(namespace)$ json='{"execute": "list_hostfwd"}'

(namespace)$ echo -n $json | nc -U /tmp/slirp4netns.sock

{"return": {"entries": [{"id": 42, "proto": "tcp", "host_addr": "0.0.0.0", "host_port": 8080, "guest_addr": "10.0.2.100", "guest_port": 80}]}}

remove_hostfwd: Remove an exposed port

(namespace)$ json='{"execute": "remove_hostfwd", "arguments": {"id": 42}}'

(namespace)$ echo -n $json | nc -U /tmp/slirp4netns.sock

{"return": {}}

Remarks:

? Client needs to shutdown(2) the socket with SHUT_WR after sending every request. i.e. No support for keep-alive and timeout.

? slirp4netns "stops the world" during processing API requests.

? A request must be less than 4096 bytes.

? JSON responses may contain error instead of return.

DEFINED NAMESPACE PATHS

A user can define a network namespace path as opposed to the default process ID:

(host)$ slirp4netns --netns-type=path ... /path/to/netns tap0

Currently, the netns-type=TYPE argument supports path or pid args with the default being pid.

Additionally, a --userns-path=PATH argument can be included to override any user namespace path defaults

(host)$ slirp4netns --netns-type=path --userns-path=/path/to/userns /path/to/netns tap0

OUTBOUND ADDRESSES

A user can defined preferred outbound ipv4 and ipv6 address in multi IP scenarios.

(host)$ slirp4netns --outbound-addr=10.2.2.10 --outbound-addr6=fe80::10 ...

Optionally you can use interface names instead of ip addresses.

(host)$ slirp4netns --outbound-addr=eth0 --outbound-addr6=eth0 ...

INTER-NAMESPACE COMMUNICATION

The easiest way to allow inter-namespace communication is to nest net? work namespaces inside the slirp4netns's network namespace.

(host)$ nsenter -t $(cat /tmp/pid) -U --preserve-credentials -n -m

(namespace)$ mount -t tmpfs none /run

(namespace)$ ip netns add foo

(namespace)$ ip netns add bar

(namespace)$ ip link add veth-foo type veth peer name veth-bar

(namespace)$ ip link set veth-foo netns foo

(namespace)$ ip link set veth-bar netns bar

(namespace)$ ip netns exec foo ip link set veth-foo name eth0

(namespace)$ ip netns exec bar ip link set veth-bar name eth0

(namespace)$ ip netns exec foo ip link set lo up

(namespace)$ ip netns exec bar ip link set lo up

(namespace)$ ip netns exec foo ip link set eth0 up

(namespace)$ ip netns exec bar ip link set eth0 up

(namespace)$ ip netns exec foo ip addr add 192.168.42.100/24 dev eth0

(namespace)$ ip netns exec bar ip addr add 192.168.42.101/24 dev eth0

(namespace)$ ip netns exec bar ping 192.168.42.100

However, this method does not work when you want to allow communication across multiple slirp4netns instances. To allow communication across multiple slirp4netns instances, you need to combine another network stack such as vde_plug(1) with slirp4netns.

(host)$ vde_plug --daemon switch:///tmp/switch null://

(host)$ nsenter -t $(cat /tmp/pid-instance0) -U --preserve-credentials -n

(namespace-instance0)$ vde_plug --daemon vde:///tmp/switch tap://vde

(namespace-instance0)$ ip link set vde up

(namespace-instance0)$ ip addr add 192.168.42.100/24 dev vde

(namespace-instance0)$ exit

(host)$ nsenter -t $(cat /tmp/pid-instance1) -U --preserve-credentials -n

(namespace-instance1)$ vde_plug --daemon vde:///tmp/switch tap://vde

(namespace-instance1)$ ip link set vde up

(namespace-instance1)$ ip addr add 192.168.42.101/24 dev vde

(namespace-instance1)$ ping 192.168.42.100

INTER-HOST COMMUNICATION

VXLAN is known to work. See Usernetes project for the example of multi-node rootless Kubernetes cluster with VXLAN: https://github.com/rootless-containers/usernetes

BESS MODE (FOR USER MODE LINUX)

slirp4netns (since v1.2.0) can be also used as a BESS-compatible server to provide network connectivity to User Mode Linux.

Terminal 1: Start slirp4netns

(host)$ slirp4netns --target-type=bess /tmp/bess.sock

Terminal 2: Start User Mode Linux

(host)$ linux.uml vec0:transport=bess,dst=/tmp/bess.sock,depth=128,gro=1 root=/dev/root rootfstype=hostfs init=/bin/bash mem=2G

(UML)$ ip addr add 10.0.2.100/24 dev vec0

(UML)$ ip link set vec0 up

(UML)$ ip route add default via 10.0.2.2

Currently, only a single instance of User Mode Linux can be connected to the slirp4netns BESS server.

See also User Mode Linux documentation: https://www.ker?nel.org/doc/html/latest/virt/uml/user_mode_linux_howto_v2.html#bess-socket-transport

BUGS

Kernel 4.20 bumped up the default value of /proc/sys/net/ipv4/tcp_rmem from 87380 to 131072. This is known to slow down slirp4netns port for?warding: https://github.com/rootless-containers/slirp4netns/issues/128.

As a workaround, you can adjust the value of /proc/sys/net/ipv4/tcp_rmem inside the namespace. No real root privi?lege is needed to modify the file since kernel 4.15.

(host)$ nsenter -t $(cat /tmp/pid) -U --preserve-credentials -n -m

(namespace)$ c=$(cat /proc/sys/net/ipv4/tcp_rmem); echo $c | sed -e s/131072/87380/g > /proc/sys/net/ipv4/tcp_rmem

SEE ALSO

network_namespaces(7), user_namespaces(7), veth(4)

AVAILABILITY

The slirp4netns command is available from https://github.com/rootless-containers/slirp4netns under GNU GENERAL PUBLIC LICENSE Version 2 (or later).