14.6.2 X11 over SSH

The scripts/designer/system/etc/ssh/sshd_conf allows X11 tunneling over ssh, e.g., from a remote VM connected to the same host-only lan as a container running the GUI application. Use `ssh -X container_ip` to enable X11 tunneling in the ssh session.

14.6.3 Traffic mirroring

Send copies of traffic from one ethernet port to another using the iptables TEE operation, e.g.,

```
iptables -t mangle -A PREROUTING -i eth1 -j TEE --gateway 172.16.3.1
```

will send copies of all incoming traffic on eth1 to the component with address 172.16.3.1. Note that gateway must be a next hop, or you will have to configure the nexthop to forward it further. This is useful for IDS labs, e.g., snort. Mirroring all incoming traffic into a component will let you reconstruct TCP sessions within that component. Mirroring output from components is not always reliable. Besides potential for duplicate traffic, Docker networks seem to sometimes gratuitously replace destination addresses with those of the Docker network gateway, i.e., the gateway to the host.

14.6.4 DNS

Install bind9 in Dockerfile. Add zone files to /etc/bind and db files to ../_system/var/cache/bind/. Add reference to the /etc/bind/named.conf.local as seen in local-dns/_bin/fixlocal.sh

14.6.5 Overriding Docker routing and DNS

Realistic network topologies require components to have /etc/resolv.conf and routing table entries that do not depend on Docker gateways and related magic. However, at some point you may want components to be able to reach the outside world. If you've fiddled resolv.conf and routing, you likely broke the default Docker method for doing this. One solution is to define an *isp* component that has a default gateway and resolv.conf as Docker defines them. Then route all traffic and DNS queries to that (making use of dnsmasq and your own resolv.conf entries). Note you will also have to set up your own NAT on that ISP component. See the dmz-example lab ISP component .local/bin/fixlocal.sh as a worked example of a simple NAT setup.

As a worked example, the dmz-example lab components (other than the ISP), typically use the .local/bin/fixlocal.sh script to delete the Docker-generated route:

```
sudo route del -host 172.17.0.1
```

And the fixlocal.sh also replaces the resolv.conf entry with either a local DNS component, or a gateway running the dnsmasq utility. The /etc/rc.local script generally sets the default gateway, and configures iptables.

14.7 User management and sudo

The Dockerfile should make the initial user, i.e., the user named in the start.config file, a member of sudoers. Otherwise, the fixlocal.sh script will not be unable to modify the environment. If desired, that user can be removed from sudoers at the end of the fixlocal.sh script.

Only the initial user (and that user's actions taken as root) are monitored. Additional users can be added, e.g., in the Dockerfile, but their actions are not monitored or recorded in artifacts.