Building a

Network Training Emulator (NTE)

Virtual Machine (VM)

version 2.0

Diarmuid O'Briain

Table of Contents

1. WHAT IS NTE 4

2. INSTALL GNU/LINUX OS 4

2.1 REMOVE UNNECESSARY PACKAGES 4
2.2 CHANGE HOSTNAME 5
2.3 ADD USER NTE 5
2.4 UPDATE SOURCES.LIST FILE 5
2.5 BEAUTIFY 6
2.6 INSTALL 6
2.7 WIRESHARK PERMISSIONS 7

3. INSTALL CORE 7

3.1 COPY FILES TO VM 7
3.2 CORE FROM SOURCE FILES 8
3.3 BUILD THE CORE MANUAL 9
3.4 LDXE ICON 10
3.5 QUAGGA TERMINAL VARIABLES 11

4. INSTALLING APPLICATIONS 12

5. RUNNING THE CORE DAEMON 13

6. RUNNING THE CORE GUI 14

6.1 RUNNING CORE GUI ON THE HOST X-SERVER 14
6.2 GNU/LINUX HOST 14
6.3 MICROSOFT WINDOWS HOST 15

7. TSHARK 17

Illustration Index

Illustration 1: NTE Desktop background 6

Illustration 2: NTE Desktop 11

Illustration 3: Running NTE GUI on host X-Server 14

Illustration 4: CORE GUI 17

1. What is NTE

The Network Training Emulator (NTE) is a network emulator from **netLabs!UG** Research Centre based on the Common Open Research Emulator (CORE). CORE was developed by a Network Technology research group, Boeing Research and Technology division. The Naval Research Laboratory is supporting further development of this open source project.

NTE uses GNU/Linux Containers (LXC) as devices linked by python programs. It is also possible to connect NTE emulated networks to live networks.

2. Install GNU/Linux OS

Install a vanilla Debian GNU/linux 8.7 i386 Operating System. It was installed with a default user called **user** who has **sudo** rights and **root** has a password **root**.

The following was choosen during the **taskselect** section:

- Debian Desktop environment:.
    - Lightweight X11 Desktop Environment (LXDE).
- SSH Server.
- Standard system utilities.

LXDE was chosen purely for performance reasons. The VMs are expected to run on basic i386 machines.

2.1 Remove unnecessary packages

Remove unnecessary packages like **LibreOffice**.

```
user@debian-i386:~$ sudo apt-get purge libreoffice-*
user@debian-i386:~$ sudo apt-get purge alsamixergui
user@debian-i386:~$ sudo apt-get purge mplayer2
user@debian-i386:~$ sudo apt-get purge lxmusic
user@debian-i386:~$ sudo apt-get purge gimp
user@debian-i386:~$ sudo apt-get purge xsane
```

2.2 Change hostname

Change the hostname to **NTE-i386**.

```
user@debian-i386:~$ sudo sed -i.bak 's/debian-i386/NTE-i386/g' /etc/hosts
user@debian-i386:~$ sudo sed -i.bak 's/debian-i386/NTE-i386/g' /etc/hostname

user@debian-i386:~$ cat /etc/hosts
127.0.0.1  localhost
127.0.1.1  NTE-i386.netlabsug.org     NTE-i386

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

user@debian-i386:~$ cat /etc/hostname
NTE-i386
```

2.3 Add user nte

Create a user **nte**.

```
user@debian-i386:~$ sudo useradd -m -s /bin/bash nte
user@debian-i386:~$ sudo passwd nte
Enter new UNIX password: nte
Retype new UNIX password: nte
passwd: password updated successfully

user@debian-i386:~$ sudo usermod -a -G sudo nte
```

Reboot the VM.

```
user@debian-i386:~$ sudo reboot
```

2.4 Update sources.list file

Check the file /**etc**/**apt**/**sources.list** file and find the element before **debian.org** as shown.

http://**<ftp.ie>**.debian.org

replace with httpredir which will allow the system to find the nearest repository.

```
nte@NTE-i386:~$ sudo sed -i.bak 's/ftp.ie/httpredir/g' /etc/apt/sources.list
```

Update the /**etc**/**apt**/**sources.list** file.

```
nte@NTE-i386:~$ sudo aptitude update
nte@NTE-i386:~$ sudo aptitude dist-upgrade
```

2.5 Beautify

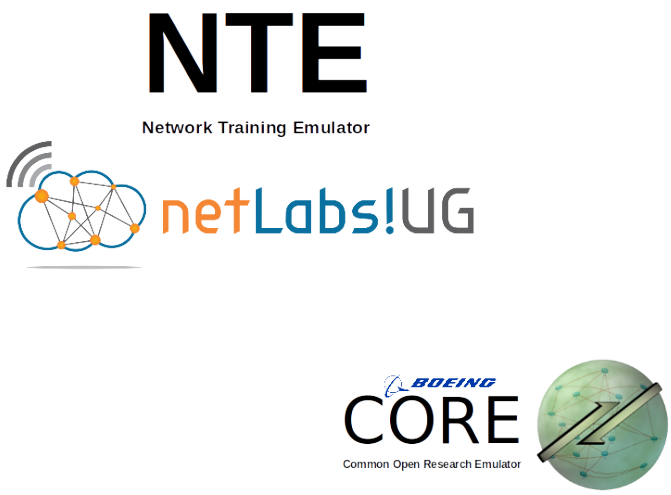

Illustration 1: NTE Desktop background

Download the following graphic files to user NTE's Pictures directory **/home/nte/Pictures**.

- http://www.netlabsug.org/NTE/nte-background.png
- http://www.netlabsug.org/NTE/core-gui.png
- http://www.netlabsug.org/NTE/netLabs!UG-small.png

Right mouse click > Desktop Preferences >

Wallpaper: ~/**Pictures/nte-background.png**
Background colour: **Preferred colour.**

2.6 Install

Install the following packages.

```
nte@NTE-i386:~$ sudo aptitude install bridge-utils
nte@NTE-i386:~$ sudo aptitude install ebtables
nte@NTE-i386:~$ sudo aptitude install iproute
nte@NTE-i386:~$ sudo aptitude install libev-dev
nte@NTE-i386:~$ sudo aptitude install tcl8.5
nte@NTE-i386:~$ sudo aptitude install tk8.5
nte@NTE-i386:~$ sudo aptitude install libtk-img
nte@NTE-i386:~$ sudo aptitude install quagga
nte@NTE-i386:~$ sudo aptitude install tcpdump
nte@NTE-i386:~$ sudo aptitude install mtr
nte@NTE-i386:~$ sudo aptitude install tshark
nte@NTE-i386:~$ sudo aptitude install wireshark
Should non-Superusers be able to capture packets? <YES>
```

It is assumed python is already installed.

2.7 Wireshark permissions

Add the group **wireshark** to the user **nte** groups, set **dumpcap** within the **wireshark** group and adjust its permissions. The **dumpcap** executable is assigned to the group instead of Wireshark itself, as **dumpcap** is responsible for all the low-level capture work. Changing its mode to 750 ensures only users belonging to its group can execute the file. Finally set file capabilities on the **dumpcap** file. **CAP_NET_ADMIN** allows various network-related operations, for example; setting privileged socket options, enabling multicasting, interface configuration, modifying routing tables. In this case it allows the setting of an interface to promiscuous mode. **CAP_NET_RAW** permits raw access to an interface for capturing directly off the wire. **=eip** grants them access in the Effective, Inheritable, and Permitted bitmaps, respectively.

CAP_NET_RAW permits raw access to an interface for capturing directly off the wire.

```
nte@NTE-i386:~$ sudo usermod -a -G wireshark nte
nte@NTE-i386:~$ sudo chgrp wireshark /usr/bin/dumpcap
nte@NTE-i386:~$ sudo chmod 750 /usr/bin/dumpcap
nte@NTE-i386:~$ sudo setcap cap_net_admin,cap_net_raw=eip /usr/bin/dumpcap
nte@NTE-i386:~$ sudo getcap /usr/bin/dumpcap
/usr/bin/dumpcap = cap_net_admin,cap_net_raw+eip
```

Logout and log back in, confirm the user **NTE** is has **wireshark** as one of its groups.

```
nte@NTE-i386:~$ id
uid=1002(nte) gid=1002(nte) groups=1002(nte),27(sudo),128(wireshark)
```

3. Install CORE

3.1 Copy files to VM

Download the CORE source files from:

http://downloads.pf.itd.nrl.navy.mil/core/source/

Upload and extract the binary files to the VM /**home**/**nte**.

```
nte@NTE-i386:~$ cd ~
nte@NTE-i386:~$ tar -xzvf core-4.8.tar.gz
nte@NTE-i386:~$ ls core-4.8/
aclocal.m4    config.h.in   doc      Makefile.am       README
bootstrap.sh  configure     gui      Makefile.in       README-Xen
Changelog     configure.ac  kernel   packaging         scripts
config        daemon        LICENSE  python-prefix.py
```

3.2 CORE from source files

To build CORE from source on Debian GNU/Linux, install these development packages.

```
nte@NTE-i386:~$ sudo aptitude install autoconf
nte@NTE-i386:~$ sudo aptitude install automake
nte@NTE-i386:~$ sudo aptitude install gcc
nte@NTE-i386:~$ sudo aptitude install make
nte@NTE-i386:~$ sudo aptitude install python-dev
nte@NTE-i386:~$ sudo aptitude install libreadline-dev
nte@NTE-i386:~$ sudo aptitude install pkg-config
nte@NTE-i386:~$ sudo aptitude install imagemagick
nte@NTE-i386:~$ sudo aptitude install help2man
```

Built the CORE package.

```
nte@NTE-i386:~$ cd core-4.8

nte@NTE-i386:~/core-4.8$ ./bootstrap.sh
Bootstrapping the autoconf system...
(Messages below about copying and installing files are normal.)
(1/4) Running aclocal...
(2/4) Running autoheader...
(3/4) Running automake...
configure.ac:124: installing 'config/compile'
(4/4) Running autoconf...

You are now ready to run "./configure".

nte@NTE-i386:~/core-4.8$ ./configure

------------------------------------------------------------------------
core 4.8 Configuration:

  Host System Type:
  C Compiler and flags:       gcc  -O3 -Werror -Wall -D_GNU_SOURCE
  Install prefix:             /usr/local
  Build GUI:                  yes
  GUI path:                   /usr/local/lib/core
  GUI config:                 ${HOME}/.core
  Daemon path:                /usr/local/sbin
  Daemon config:              /etc/core
  Python install prefix:      /usr/local
  Python modules:             ${pyprefix}/lib/python2.7/dist-packages
  Logs:                       /var/log

Features to build:
  Python bindings:            yes
  Linux Namespaces emulation: yes
  FreeBSD Jails emulation:    no
  Documentation:              yes

------------------------------------------------------------------------
On this platform you should run core-gui as a normal user.
------------------------------------------------------------------------
```

```
nte@NTE-i386:~/core-4.8$ make -j8
nte@NTE-i386:~/core-4.8$ sudo make install
```

After this step, the following are available.

```
/etc/init.d/core               - SysV init script to start daemon
/usr/local/sbin/core-daemon    - Core daemon to run in the background
/usr/local/bin/core-gui        - Graphical tool

/usr/local/sbin/core-daemon
/usr/local/bin/core-gui
/usr/local/sbin/[vcmd, vnoded, coresendmsg, core-cleanup.sh]
/usr/local/lib/core/*
/usr/local/share/core/*
/usr/local/lib/python2.6/dist-packages/core/*
/usr/local/lib/python2.6/dist-packages/[netns,vcmd].so
/etc/core/*
/etc/init.d/core
```

core-daemon - CORE daemon manages emulation sessions

core-gui - CORE graphical user interface

3.3 Build the CORE manual

Built the CORE Manual documentation from the doc/ directory.

```
nte@NTE-i386:~/core-4.8$ sudo aptitude install python-sphinx
nte@NTE-i386:~/core-4.8$ cd doc
nte@NTE-i386:~/core-4.8/doc$ make html
Build finished. The HTML pages are in _build/html.
```

3.4 LDXE Icon

Copy the CORE GUI icon to the LXDE pixmaps directory.

```
nte@NTE-i386:~$ sudo cp ~/Pictures/core-gui.png /usr/share/pixmaps/

nte@NTE-i386:~$ ls -la /usr/share/pixmaps | grep core
-rw-r--r--   1 root root  2368 Feb 14 07:05 core-gui.png

nte@NTE-i386:~$ sudo bash -c 'cat << EOF >
/usr/local/share/applications/core-gui.desktop
[Desktop Entry]
Version=1.0
Type=Application
Name=NTE
GenericName=NTE-GUI
Comment=GNU/Linux based Network Training Emulator
Encoding=UTF-8
Exec=/usr/local/bin/core-gui
Icon=/usr/share/pixmaps/core-gui.png
Terminal=false
StartupNotify=false
Categories=Network;Internet;
EOF'
```

lxpanel processes must be killed before it can reload an *lxpanel* profile. Then find and delete the cached menu items to ensure updates appear.

```
nte@NTE-i386:~$ sudo killall lxpanel
nte@NTE-i386:~$ find ~/.cache/menus -name '*' -type f -print0 | xargs -0 rm
```

Reboot the system to reload the panel.

```
nte@NTE-i386:~$ sudo shutdown -r now
```

In LXDE Menu icon > Internet > NTE Network Emulator

Right mouse click and select "**Add to Desktop**"

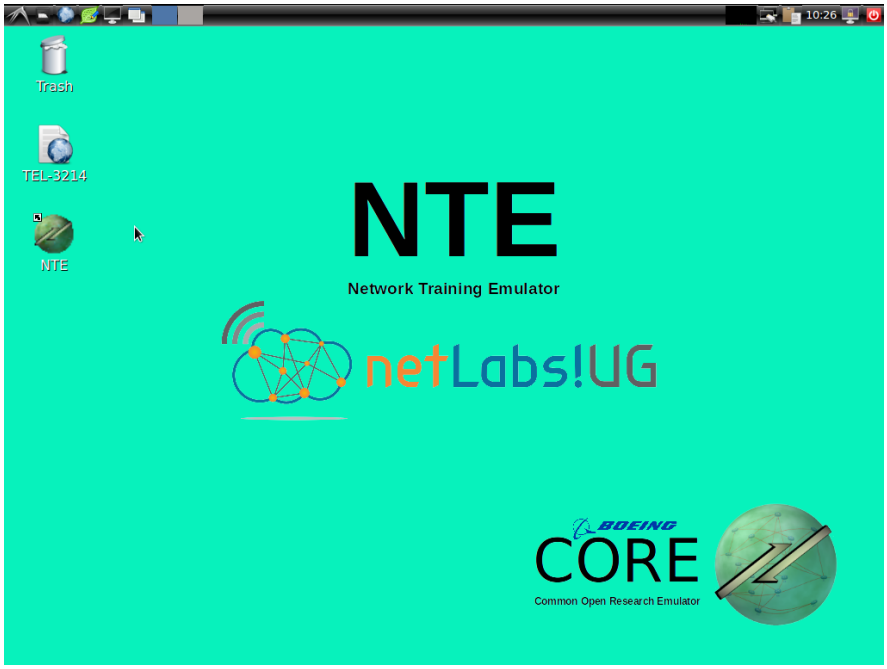

Illustration 2: NTE Desktop

3.5 Quagga Terminal variables

Add the following variables which are required by Quagga vtysh shell.

```
nte@NTE-i386:~$ sudo bash -c 'cat << EOF >> /etc/environment

# Added for Quagga in NTE
VTYSH_PAGER=more
EOF'

nte@NTE-i386:~$ cat /etc/environment | grep VTYSH_
VTYSH_PAGER=more

nte@NTE-i386:~$ sudo bash -c 'cat << EOF >> /etc/bash.bashrc

# Added for Quagga in NTE
export VTYSH_PAGER=more
EOF'

nte@NTE-i386:~$ cat /etc/bash.bashrc | grep VTYSH_
export VTYSH_PAGER=more
```

Logout and back in again to enable these variables.

4. Installing applications

Applications that may be needed for simulations should be installed.

```
nte@NTE-i386:~$ sudo aptitude install isc-dhcp-server
nte@NTE-i386:~$ sudo aptitude install vsftpd
nte@NTE-i386:~$ sudo aptitude install apache2
nte@NTE-i386:~$ sudo aptitude install radvd
nte@NTE-i386:~$ sudo aptitude install openvpn
nte@NTE-i386:~$ sudo aptitude install racoon
nte@NTE-i386:~$ sudo aptitude install mgen
nte@NTE-i386:~$ sudo aptitude install bird
nte@NTE-i386:~$ sudo aptitude install iperf
nte@NTE-i386:~$ sudo aptitude install fwbuilder
nte@NTE-i386:~$ sudo aptitude install ufw
nte@NTE-i386:~$ sudo aptitude install conntrack
nte@NTE-i386:~$ sudo aptitude install lynx
nte@NTE-i386:~$ sudo aptitude install gnupg2
nte@NTE-i386:~$ sudo aptitude install zip
nte@NTE-i386:~$ sudo aptitude install python-pip
nte@NTE-i386:~$ sudo aptitude install fping
nte@NTE-i386:~$ sudo pip install netifaces
```

5. Running the CORE daemon

Debian 8.7 uses the **systemd** init system and NTE 4.8 installs an /**etc**/**init.d**/**core-daemon** init script. **systemd** must be made aware of the new start script so it can treat it as a service. **systemd daemon-reload**, scans for new or changed units.

```
nte@NTE-i386:~$ sudo systemctl daemon-reload
```

Enable the core-daemon to start at boot-time.

```
nte@NTE-i386:~$ sudo systemctl enable core-daemon
Synchronizing state for core-daemon.service with sysvinit using update-
rc.d...
Executing /usr/sbin/update-rc.d core-daemon defaults
Executing /usr/sbin/update-rc.d core-daemon enable
```

Start the core-daemon.

```
nte@NTE-i386:~$ sudo systemctl start core-daemon
Setting up user config area /home/core/.core, /home/core/.core/configs,
and /home/core/.core/myservices
Creating a default /home/core/.core/nodes.conf
Creating a default /home/core/.core/servers.conf
Creating a default /home/core/.core/plugins.conf
Connecting to "core-daemon" (127.0.0.1:4038)...connected.
```

Check the status of the core-daemon.

```
nte@NTE-i386:~$ systemctl status core-daemon
● core-daemon.service - LSB: Start the core-daemon NTE daemon at boot time
   Loaded: loaded (/etc/init.d/core-daemon)
   Active: active (running) since Thu 2016-02-18 17:10:19 GMT; 12min ago
 Main PID: 2510 (python)
   CGroup: /system.slice/core-daemon.service
           └─2510 /usr/bin/python /usr/local/sbin/core-daemon -d
```

6. Running the CORE GUI

Start the **core-gui** as a user by either clicking on the icon on the desktop or running the command:

```
nte@NTE-i386:~$ core-gui
```

However to do this requires the VM to emulate the graphics. It is better to SSH to the VM and use X11 redirection ro run the GUI on the X-Server of the host computer.

6.1 Running CORE GUI on the Host X-Server

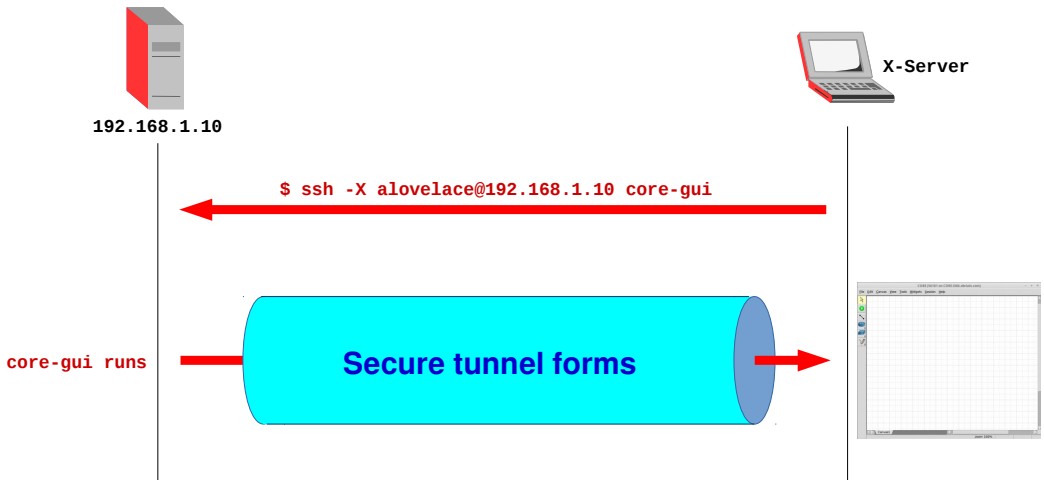

Illustration 3: Running NTE GUI on host X-Server

It is quite beneficial to run the core-gui on the host computer's X-Server. This means that the VM does not have to render the graphical output to the virtual graphics card for the VM that is created in software but can render it directly on the host which uses the actual graphics card.

This in demonstated in Illustration 3 where the user on the host computer connects to the VM via SSH with the special -X switch which enables X11 forwarding. Any graphical application ran on the VM will be redirected through the SSH tunnel to the host to be displayed using the hosts own X-Server.

6.2 GNU/Linux Host

SSH to the VM guest using the -X switch for X11 redirection. The core-gui graphics will be redirected to the GNU/linux Host X-Server.

```
alovelace@riomhaire:~$ ssh -X nte@192.168.10.4 core-gui
```

6.3 Microsoft Windows Host

To run the **core-gui** on a Microsoft Windows Host it is necessary to install an X-Server. As it does not have one by default. This can be downloaded from http://www.straightrunning.com/xmingnotes.

- 64-bit
  - **rel_x64_Xming-7-7-0-4-setup.exe**
  - **rel_x64_Xming-portablePuTTY-7-7-0-4-setup.exe**
- 32-bit
  - **rel_Xming-7-7-0-4-setup.exe**
  - **rel_Xming-portablePuTTY-7-7-0-4-setup.exe**
- **Xming-fonts-7-5-0-93-setup.exe**

Determine the OS architecture, 32-bit or 64-bit and download and install the two files appropriate files for the system. The first is the X-Server and the second is an adapted PuTTY to give SSH tools.

Install the final fonts file, it is common for both 32-bit and 64-bit Xming. During the install a message about there already being an Xming directory will be given, ignore the error and install into the Xming directory. It will create a fonts directory within Xming.

Adjust the Microsoft Windows PATH to incorporate the Xming directory.

- Click ***Start***, right-click on Computer, select ***Properties***
- ***Advanced System Settings***
- ***Environment Variables***
- ***System Variables***
- Scroll to find **PATH**, highlight and click **Edit**.
- Add the following to the Variable value: and click **OK**

```
;C:\Program Files\Xming
```

- Click OK.

If a command shell is open, it will need to be closed and re-opened to re-read variables.

6.3.1 Running the X-Server on Microsoft Windows

Immediately after install the X-Server will be running and following this step will resilt in an error. Simply ignore the error, it is simply informing that the X-Server is already running.

If however it is not running, the following command will start the X-Server as a background process.

```
C:\> xming -multiwindow -clipboard
```

-multiwindow - Starts the integrated Microsoft Windows-based window manager, which launches each top-level X Window in its own Microsoft Windows window.

-clipboard - Enables the integration between the X11 clipboard and Microsoft Windows clipboard. The default is disabled.

6.3.2 Connect to the VM and run the core-gui application

Much like with the GNU/Linux host, SSH to the GNU/Linux Server in the VM and run the application. It is redirected to the X-Server running on the Microsoft Windows computer via X11 forwarding.

```
C:\> plink -ssh -2 -X <user>@<GNU/Linux hostname | IP address> <application>
```

i.e.

```
C:\> plink -ssh -2 -X nte@192.168.1.252 core-gui
```

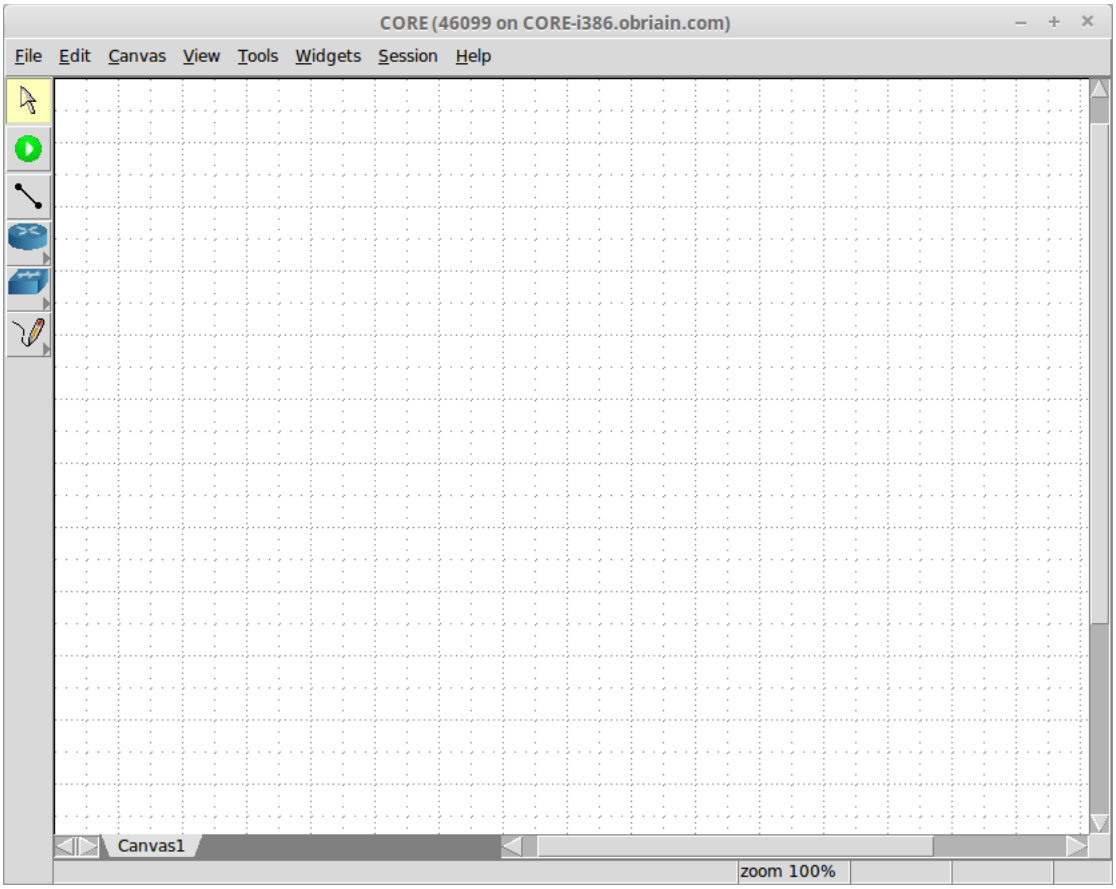

Illustration 4: CORE GUI

7. Tshark

Tshark runs but displays the following message. This is OK because **tshark** is running as root within each **LinuX Container (LVC)**.

```
  tshark: Lua: Error during loading:
   [string "/usr/share/wireshark/init.lua"]:46: dofile has been disabled due
to running
   Wireshark as superuser. See
http://wiki.wireshark.org/CaptureSetup/CapturePrivileges
   for help in running Wireshark as an unprivileged user.
  Running as user "root" and group "root". This could be dangerous.
  Capturing on 'eth0'
```

This page is intentionally blank