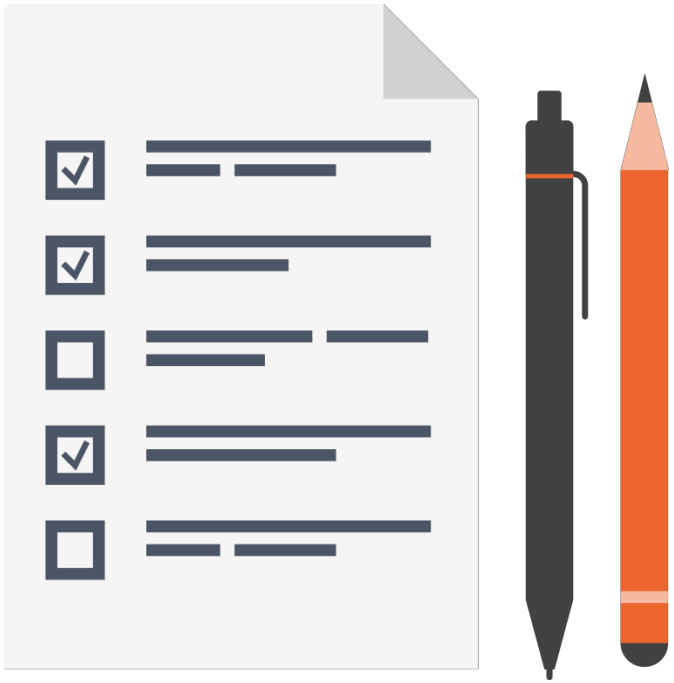
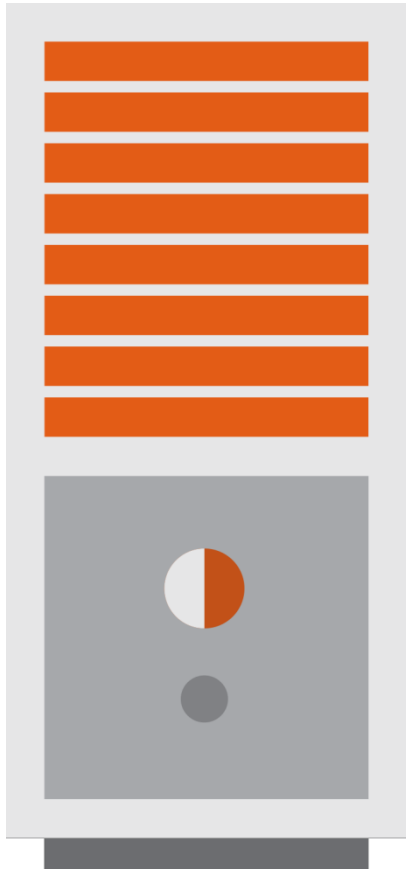
Advanced Network Administration

Module Overview



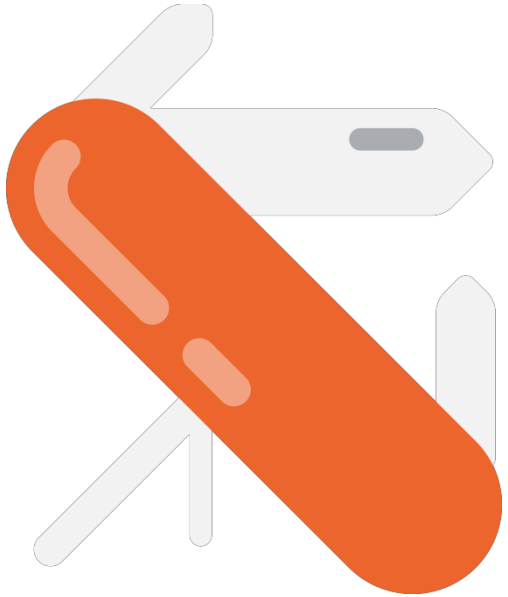
- Port and traffic analysis
- Network troubleshooting
- Device naming schemes
- NetworkManager

Used in This Module



As **RHEL 7.1** is used predominantly in this module as we look at the new device naming schemes implemented in Red Hat 7.1

Advanced Network Configuration



Command line tools

- tcpdump
- netstat
- lsof
- nc
- nmap

tcpdump

Command line packet analyser. Many of us will have heard of the graphical Wireshark, (Ethereal), packet capture tool; however from the command line we too can interrogate packets.

```
$ sudo tcpdump -i eth0
$ sudo tcpdump -c 5 -i eth0
$ sudo tcpdump -i eth0 not port 22
```

Capturing Network Traffic with tcpdump

Display all packets captured on eth0

Only capture 5 packets

Don’t capture SSH traffic, useful when remoting into the host

Demo: Using tcpdump

netstat

Commonly used to display open ports on a host and is found on Unix, Linux and Windows hosts

```
$ netstat -a
$ netstat -nr
$ netstat -nalt
$ netstat -lx
$ ls -l /tmp/filexxx
```

```
-a Show all
-n Numeric
-r Routes
-l Listening
-t TCP
-x Sockets
```

Using netstat

Display a long listing on one of the files listed in the -x results will confirm it is a socket

```
$ sudo lsof -i
$ sudo lsof -iTCP:22
$ sudo lsof -i@192.168.40.3
```

Similarly lsof can list connections

Option -i with no arguments will list all open ports

Then we show TCP and port 22 only

Finishing with connections to or from the host 192.168.40.3

Demo: Using netstat and lsof

nc

The command nc or netcat is a very useful command for creating and testing network connections

```
$ nc 192.168.0.4 80
GET
```

Test a web server

We connect to the web server with **nc** and then issue the **GET** command

Server 1

```
$ nc -l 8888
```

Server 2

```
nc server1 8888
```

Chat window between two hosts

With the connection in place you can “chat” between the hosts

Server 1 is listening on port 8888

Server 1

```
$ tar -czf h.tgz /etc/hosts
$ nc server2 8888 < h.tgz
```

Server 2

```
$ nc -l 8888 > hosts.tgz
```

Transfer files between hosts

Here Server 2 is listening on port 8888 and the file is sent from Server 1

Demo: Using nc

nmap

Nmap or the "**Network Mapper**" is used for network explorations and security auditing; being able to scan large networks or single hosts identifying open ports and operating system versions

```
$ nmap --iflist
```

Display local ports and routes

If nmap is not installed it can be installed:

sudo yum install nmap

sudo apt-get install nmap

```
$ nmap 192.168.0.4
$ sudo nmap 192.168.0.4
```

```
[andrew@redhat7 ~]$ nmap 192.168.0.4

Starting Nmap 6.40 ( http://nmap.org ) at 2015-01-05 10:45 GMT
Nmap scan report for 192.168.0.4
Host is up (0.020s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

```
[andrew@redhat7 ~]$ sudo nmap 192.168.0.4
[sudo] password for andrew:

Starting Nmap 6.40 ( http://nmap.org ) at 2015-01-05 10:56 GMT
Nmap scan report for 192.168.0.4
Host is up (0.0073s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: B8:27:EB:9A:B8:15 (Raspberry Pi Foundation)
```

Simple scans…are not so simple

As a standard user a TCP scan is initiated: nmap -sT 192.168.0.4

As a privileged account a SYN scan is used: nmap -sS

TCP Scan

TCP scan can be initiated by standard users and use the UNIX connect() libraries. As such a full TCP 3 way handshake is initiated. These scans are easy to detect and mitigate against.

SYN Scan

The SYN can requires root privileges on the system using nmap. The connection is broken down on the receipt of a SYN packet from the target port. Modern firewalls can detect SYN scans but it is made more difficult in the way nmap alters timings.

```
$ nmap -sV 192.168.0.4
```

```
[andrew@redhat7 ~]$ nmap -sV 192.168.0.4

Starting Nmap 6.40 ( http://nmap.org ) at 2015-01-05 11:32 GMT
Nmap scan report for 192.168.0.4
Host is up (0.019s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
80/tcp open  http    Apache httpd 2.2.22 ((Debian))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Version scans

The version of the service hosting a port can be detected with **-sV**

$ nmap -A 192.168.0.4

OS Detection

One can be a little aggressive using the **-A** option which includes OS detection

```
[andrew@redhat7 ~]$ sudo nmap -A 192.168.0.4

Starting Nmap 6.40 ( http://nmap.org ) at 2015-01-05 11:44 GMT
Nmap scan report for 192.168.0.4
Host is up (0.0042s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
| ssh-hostkey: 1024 ae:6d:48:58:c1:50:8c:b9:4c:e4:1b:13:bf:96:ea:8b (DSA)
| 2048 20:a9:a9:8f:a5:06:0c:6f:31:23:9d:5f:cd:ca:d0:7f (RSA)
|_256 cb:a8:5b:67:88:2e:b3:f6:ed:2d:d0:16:2a:5d:17:4f (ECDSA)
80/tcp open  http    Apache httpd 2.2.22 ((Debian))
|_http-title: Test
MAC Address: B8:27:EB:9A:B8:15 (Raspberry Pi Foundation)
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=6.40%E=4%D=1/5%OT=22%CT=1%CU=44204%PV=Y%DS=1%DC=D%G=Y%M=B827EB%TM
OS:=54AA793F%P=x86_64-redhat-linux-gnu)SEQ(SP=106%GCD=1%ISR=10C%TI=Z%CI=I%I
OS:I=I%TS=7)SEQ(SP=106%GCD=1%ISR=10C%TI=Z%TS=7)SEQ(SP=106%GCD=1%ISR=10C%TI=
OS:Z%CI=I%TS=7)OPS(O1=M5B4ST11NW6%O2=M5B4ST11NW6%O3=M5B4NNT11NW6%O4=M5B4ST1
OS:1NW6%O5=M5B4ST11NW6%O6=M5B4ST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7
OS:120%W6=7120)ECN(R=Y%DF=Y%T=40%W=7210%O=M5B4NNSNW6%CC=Y%Q=)T1(R=Y%DF=Y%T=
OS:40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%
OS:O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=4
OS:0%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%
OS:Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=
OS:Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   4.20 ms 192.168.0.4

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.07 seconds
```

```
$ nmap -p80 192.168.0.4
$ nmap -p80 --script http-title 192.168.0.4
$ nmap -p80 --script http-enum 192.168.0.4
$ nmap -p80 --script http-enum \
  --script-args http-enum.displayall 192.168.0.4
```

Working with HTTP

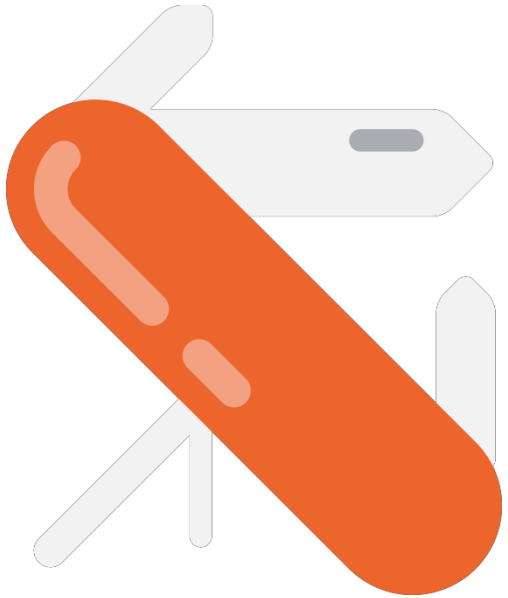
If we only want to check the one port then use **-p80**, it will be quicker

Using NSE scripts we can start to gain more information

Arguments to scripts can show even more

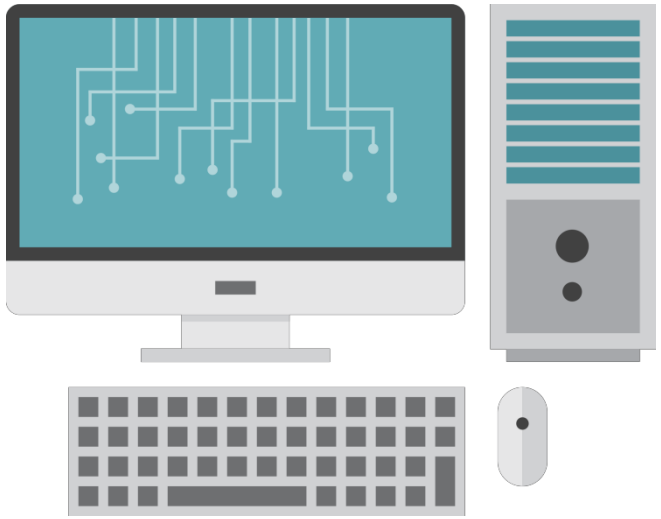
Demo: Network Auditing with nmap

Troubleshooting Network Issues

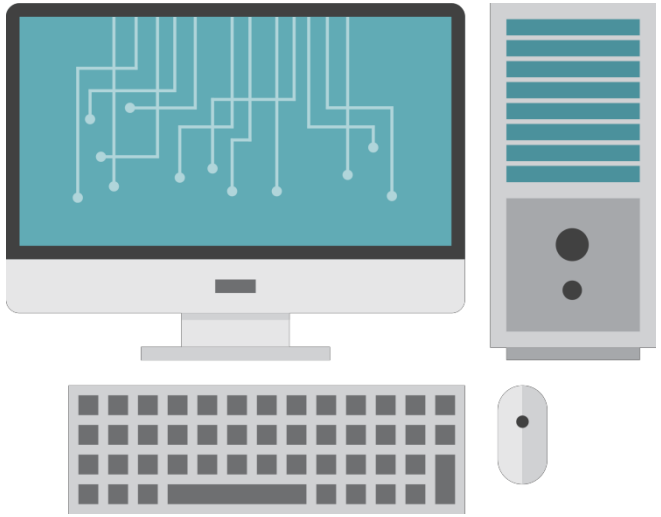


Network troubleshooting tools

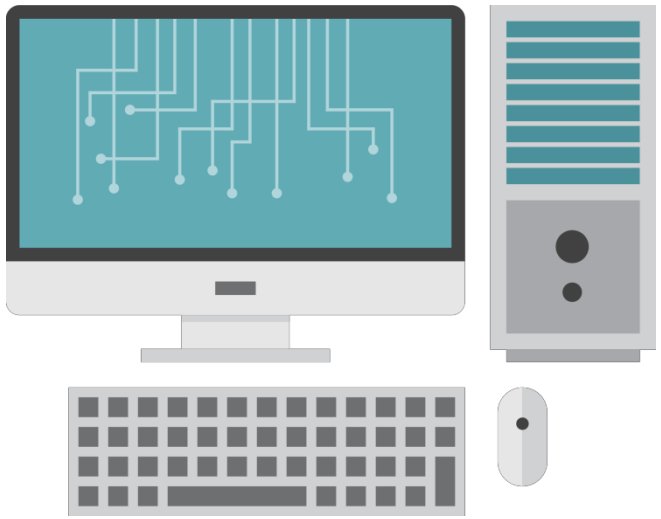
- ip / ifconfig / route
- iptables -L
- ping
- dig , /etc/hosts, nsswitch.conf, resolv.conf
- /etc/hosts.allow + deny
- Log files



- Check we have an IP Address
- The correct subnet mask
- The correct default gateway



- Check blocked ports with the Firewall
- $ sudo iptables -L
- $ sudo iptables -F
- Use nmap to check ports are open



- Check name resolution
- Order of the lookup in **/etc/nsswitch.conf**
- Use **dig** or **host** to check name entries
- Check **/etc/resolv.conf** along with **ifcfg-** files

TCP Wrappers

```
ldd </path/to/service name> | grep libwrap
```

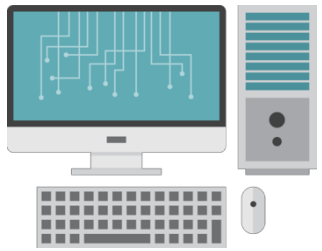
/etc/hosts.allow

in.tftpd : 192.168.0.3

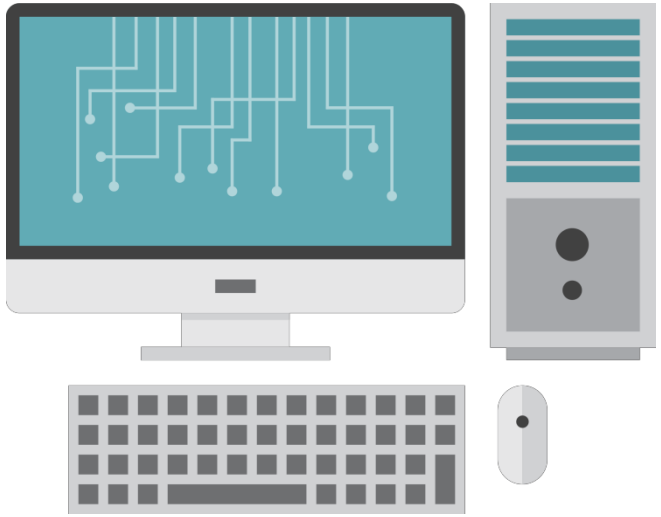
/etc/hosts.deny

in.tftpd : ALL

If the client appears in both files then the allow file takes precedence and access is granted; in this case 192.168.0.3 would have access to TFTP



If a service supports TCP Wrappers then access may be restricted via the hosts.allow or hosts.deny files



- Did you check the logs yet?
- /var/log/messages **or** /var/log/syslog
- /var/log/audit/audit.log
- dmesg

Demo: Network Troubleshooting

Managing Hostnames

Hostnames

The persistent hostname of a system is stored in:

/etc/HOSTNAME - SUSE

/etc/hostname - Debian

/etc/sysconfig/network - RHEL 6

/etc/hostname & /etc/machine-info - RHEL 7

```
$ hostname
$ sudo hostname alexis
```

hostname

The **transient** hostname set from the hostname file and can be displayed using the hostname command

The root user can change the transient name whilst the system is running but will **NOT** persist reboots

```
[andrew@redhat7 ~]$ hostnamectl
   Static hostname: redhat7.tup.com
   Pretty hostname: Red Hat 7.tup.com
         Icon name: computer
           Chassis: n/a
        Machine ID: eeec69a9c3d64634b3be4fc877d17cae
           Boot ID: 1b44f70f08e24da48d411ea93a60e214
  Operating System: Red Hat Enterprise Linux Server 7.1 (Maipo)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:7.1:beta:server
            Kernel: Linux 3.10.0-210.el7.x86_64
      Architecture: x86_64
[andrew@redhat7 ~]$
```

```
$ hostnamectl
$ hostnamectl set-hostname "Red Hat 7.tup.com"
```

And in RHEL 7……..

The hostname can be controlled via **systemd** using **hostnamectl**

The permissions to set the name are controlled via the **Policy Kit** and members of the **wheel** administrative group can set the name without the need of sudo. This sets both the persistent and transient names and if required the **pretty name**

Demo: Using hostnamectl

Red Het Enterprise Linux 7 Network Device Naming Schemes

Consistent Naming Scheme RHEL 7

With more and more LoM (LAN on Motherboard) and other multi-port LAN adaptors taking to the market place the previously inconsistent naming scheme: eth0, eth1, et cetera has been replaced with the **Consistent Naming Scheme**. This is where the address of the card is used it's name. The naming is managed by **systemd**

```
Starting Nmap 6.47 ( http://nmap.org ) at 2015-01-05 15:18
GMT
**************************************INTERFACES***************************
DEV          (SHORT)    IP/MASK
enp9s0       (enp9s0)   (none)/0
wlp12s0      (wlp12s0)  192.168.0.158/24
```

Consider the output from nmap --iflist

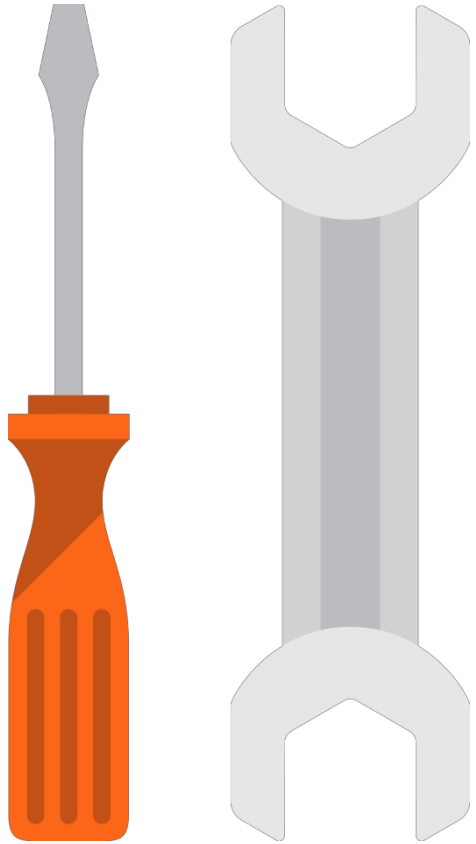
The device **enp9s0** is EN for Ethernet, P for PCI in bus 9 slot 0

The device **wlp12s0** is WL for Wireless LAN and PCI bus 12 slot 0

```
$ lspci | grep 09:00.0
$ lspci | grep 0c:00.0
```

This addresses can be ratified using lspci

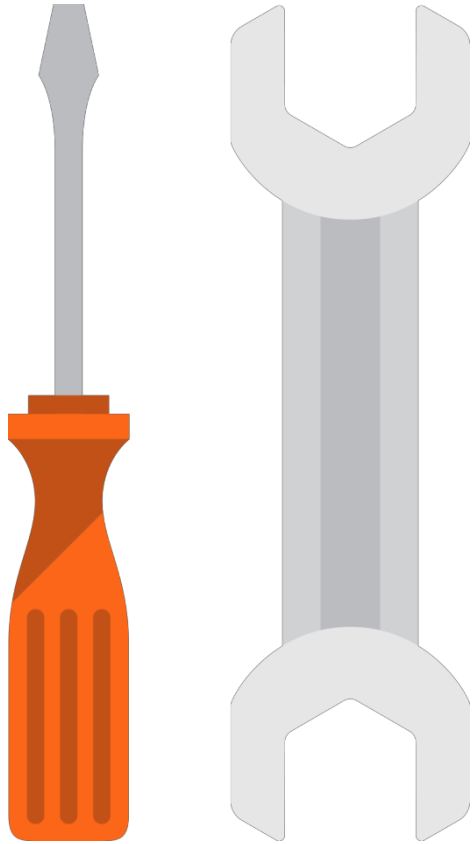
Note for PCI Bus 12 we search for 0c as the lspci output is in HEX



Disabling Consistent Network Device Naming

Method 1

Add the HWADDR attribute to network script file and either rename the file ifcfg-eth0 or configure the DEVICE name attribute



Disabling Consistent Network Device Naming

Method 2

Edit /etc/default/grub

GRUB_CMDLINE_LINUX="crashkernel=auto **biosdevname=0 net.ifnames=0** quiet“

grub2-mkconfig -o /boot/grub2/grub.cfg

Demo: Setting Inconsistent Network Device Names

NetworkManager

NetworkManager

The NetworkManager service appears in many different Linux distributions including RHEL 7 and Ubuntu 14.04. As well as the obvious access to the GUI to ease connection to WiFi it can provide an easy mechanism to store many configurations per interface.

Interacting with NetworkManager RHEL 7

- sudo systemctl status NetworkManager.service
- Control Center
- nmtui
- nmcli

```
$ nmcli device wifi list
```

Listing WiFi Networks with nmcli

```
$ sudo nmcli connection add con-name wired-home \
ifname enp9s0 type ethernet ip4 192.168.0.8/24 gw4
192.168.0.1
$ sudo nmcli connection modify wired-home ipv4.dns
“192.168.0.3 8.8.8.8”
$ nmcli -p connection show wired-home
```

Creating Connection Profiles with nmcli

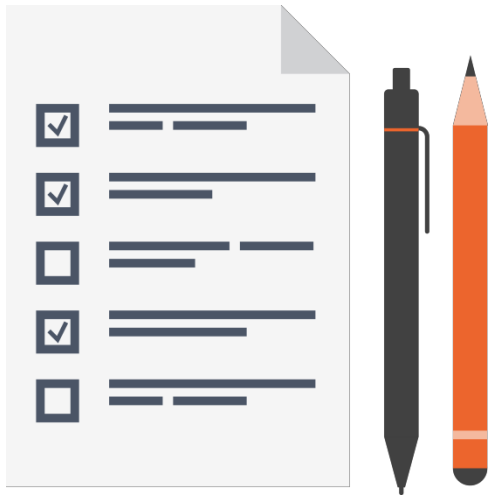
We can add new connections profiles

We need to edit the connection profile to add DNS information

Use show to display the connection profile

Demo: Using nmcli

Summary



- Used port monitoring and packet capture tools
- Troubleshooting networks
- RHEL 7 Consistent Network Device Names
- Hostnames
- NetworkManager service