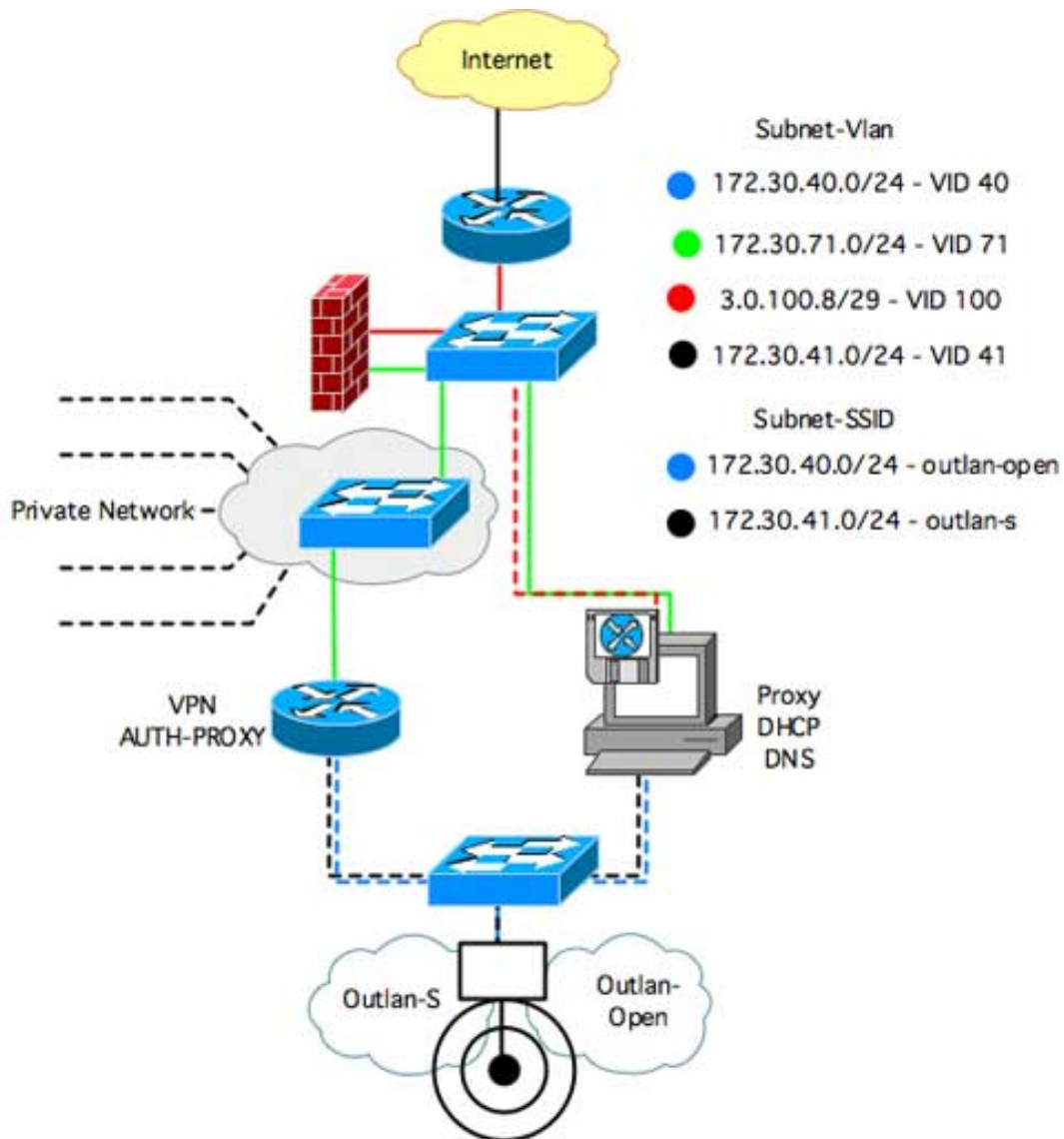


Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

Our router expert continues his series on building a secure WLAN with a Linux base. This article covers standard interface configuration of the http proxy server.

To provide the network access and security functionality needed for our WLAN environment, the Web proxy server requires four IP network connections. These connections can be discrete, requiring one physical interface per network connection or over one or more physical interfaces using 802.1q VLAN interfaces. Linux, of course, can support both approaches. In order to sufficiently cover the commands, functionality and processes related to Linux networking, we are going to cover Linux networking in two sections. This month will cover "standard" interface configuration. Next month will cover IP routing and VLAN configuration. But before we dive into Linux networking, let's quickly review what our WLAN/LAN network configuration looks like and the access controls used to ensure security.



The WLAN is partitioned from the LAN using two devices. A Cisco IOS router provides filtered IPsec VPN client access to resources on the private data LAN. The WLAN proxy server provides proxied Web access, along with DHCP and DNS services for the WLAN segments. To ensure the security of the LAN, the WLAN is segmented into two discrete wireless segments. RF connectivity to these

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

discrete segments can be provided by a single "smart" access point (AP) with support for multiple SSIDs and VLANs (such as a Cisco 1200) or by two "dumb" APs, each connected to one WLAN segment using two different RF channels. Using two SSIDs makes it possible to segment WLAN access into two service models. The SSID called "outlan-s" provides IPsec protected access to private LAN resources for known users. The SSID "outlan-open" provides secured Internet Web access to anonymous users.

The known users connecting to the SSID outlan-s require two predefined credentials in order to join the WLAN:

- The user requires a WEP key or WAP pre-shared key to connect to the AP.
- The IP address is assigned by DHCP and is bound to the user's host network MAC address.
- Once connected to the SSID outlan-s, users have access to the Web (via the proxy server) only.

Additional access to resources on the private LAN requires two additional predefined credentials:

- A user account on the VPN gateway that provides a 3DES secure link over the WLAN (in addition to WEP or WAP).
- A user account on the firewall that defines which private resources the user can access.

The users connecting to the SSID outlan-open have only one predefined credential requirement, which is a user account to connect to the Web proxy server. Once connected, they only have Web access. Now that we've reviewed, let's move on to Linux networking.

Linux network interface configuration uses separate commands for Layer 2 and Layer 3 configuration.

The proxy server's Layer 2 configurations -- i.e., port speed and duplex settings -- can be managed using one of two commands:

```
/sbin/ethtool  
/sbin/mii-tool
```

Of the two, /sbin/ethtool is the more feature-rich, while /sbin/mii-tool is the elder and more limited of the set. Both commands, however, are equally adequate to perform the basic Layer 2 configuration tasks.

To get an interface's current configuration status:

```
/sbin/ethtool {eth*}  
/sbin/mii-tool -v {eth*}
```

To get an interface's driver information (/sbin/ethtool only):

```
/sbin/ethtool -i {eth*}
```

To set an interface's port speed:

```
/sbin/ethtool -s {eth*} speed {10|100|1000}
```

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

```
/sbin/mii-tool -a {100baseTx-FD | 100baseTx-HD | 10baseT-FD | 10baseT-HD} {eth*}
```

To set an interface's duplex mode:

```
/sbin/ethtool -s {eth*} duplex {half|full}
```

```
/sbin/mii-tool -a {100baseTx-FD | 100baseTx-HD | 10baseT-FD | 10baseT-HD} {eth*}
```

To enable or disable an interface's auto-negotiation mode:

```
/sbin/ethtool -s eth1 autoneg {on | off}
```

To force an interface to re-auto-negotiate speed and duplex with the connected switch:

```
/sbin/ethtool -r {eth*}  
/sbin/mii-tool -r {eth*}
```

The two commands offer additional capabilities for monitoring, testing and pulling more esoteric operational information. But for all practical purposes, the server's speed and duplex settings are an administrator's primary concern.

In the case of workstations, the final result of the Ethernet auto-negotiation is relative in terms of impact. In most cases, auto-negotiation will result in the most optimum rate for the transport environment. (This does not necessarily mean that the rate will be optimum for the user.) In the case of servers, however, leaving the network connection to chance is not the best approach. Server Layer 2 interfaces should be hard set to meet the performance and functionality needs of the server and its applications.

The importance of hard setting network speed and duplex options is far greater in Linux environments than in Windows environments. Because of the open-source basis for the platform, the quality of network drivers and the compatibility with auto-negotiation varies. The interface is intended to select the fastest mutually supported media technology, but in practice this does not always hold true. It should also be said that network hardware vendors have just as many problems with auto-negotiation. So the safest bet with servers is to hard set the values on the switch and host. That can be accomplished during the boot process with the addition of an rc script or at the command line once the server has completed the boot process.

When it comes to Layer 3 configuration, as is the case with Layer 2, there are two commands available:

```
/sbin/ifconfig  
/sbin/ip
```

The `/sbin/ifconfig` command is the de facto standard interface configuration command for Linux. The `ifconfig` command is used to enable or disable a kernel-resident network interfaces, as well as setting the interface's IP address and sundry information. The `/sbin/ip` command is a newcomer alternative to `/sbin/ifconfig`. Its support varies depending on Linux distribution. The `/sbin/ip` command not only supports IP interface configuration, but also supports Layer 2 configuration along with support for defining IP routing information. Think of it as the Superman of network configuration commands. As I

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

said, support varies depending on distribution, but the source is also available on the Web. Here is the basic command syntax for configuring IP using `/sbin/ifconfig` and `/sbin/ip`:

To see the interfaces known to the kernel:

```
/sbin/ifconfig -a  
/sbin/ip link
```

To see information on active interfaces:

```
/sbin/ifconfig  
/sbin/ip addr
```

To configure host interface IP address:

```
/sbin/ifconfig {eth*}{ip address in dotted-quad} netmask (subnet mask in dotted-quad)
```

```
/sbin/ip addr add {ip address in dotted-quad/subnet prefix (8-30)} dev {eth*}
```

To bring up an interface:

```
/sbin/ifconfig {eth*} up  
/sbin/ip link set {eth*} up
```

To bring down an interface:

```
/sbin/ifconfig {eth*} down  
/sbin/ip link set {eth*} down
```

To set the maximum transmission unit on an interface, add:

```
/sbin/ifconfig mtu {1-1500}  
/sbin/ip link set {eth*} mtu {1-1500}
```

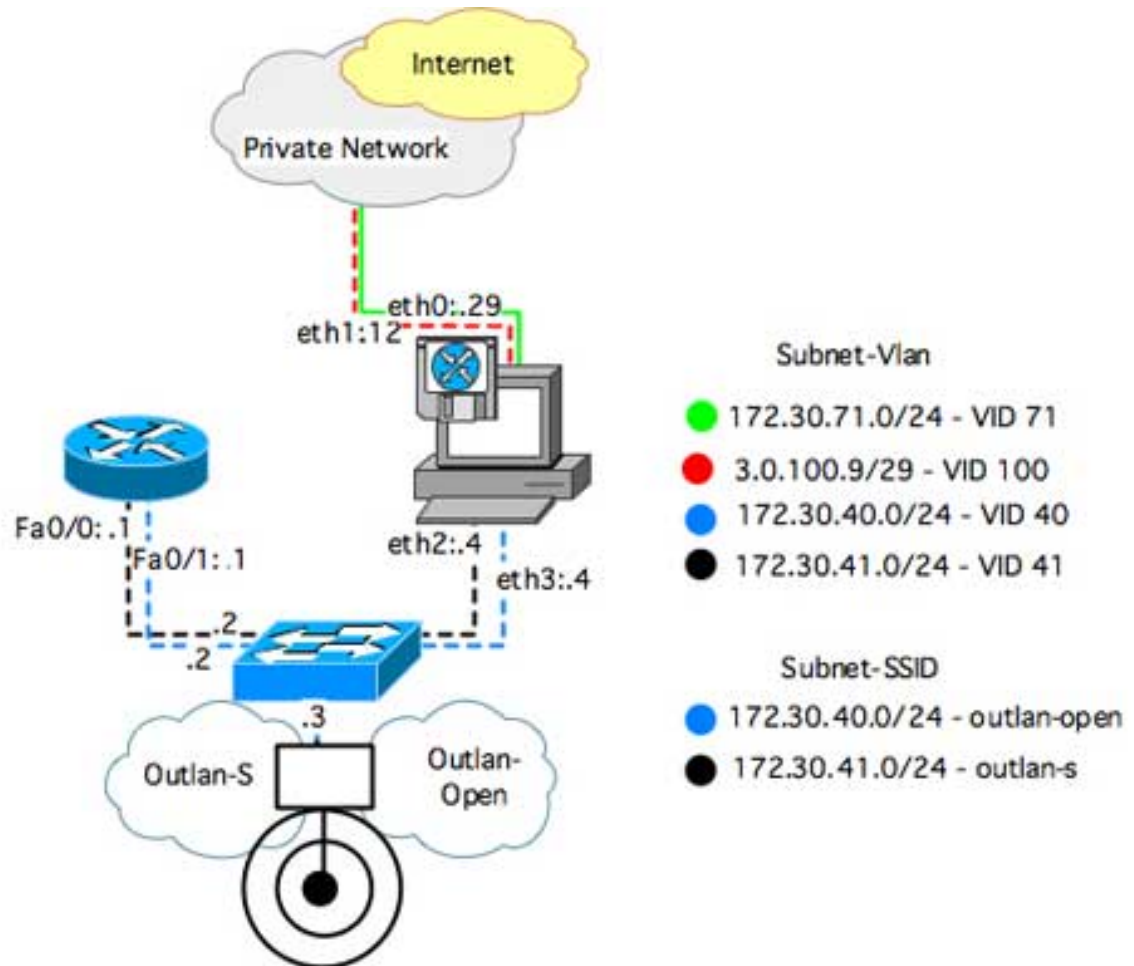
Once a Linux server has an active IP interface (on a server with only one interface), all that remains to be configured is the IP default gateway (DG). The DG is set using the `/sbin/route` or `/sbin/ip route` command using the following syntax:

```
/sbin/route add default gw {ip gateway address in dotted-quad}  
/sbin/ip route add default via {ip gateway address in dotted-quad}
```

Once the DG is configured, the server can reach hosts outside of its local IP subnet. At this point, we have reviewed all of the commands needed to configure the proxy server's network interfaces from a command shell or using a shell script.

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin



To configure the proxy server's interfaces, each 100 Mbps full duplex addressed for the network topology above the configuration script would look something like this:

```
#!/bin/sh
# Simple shell script to configure speed, duplex and IP addressing
# for WLAN proxy server interfaces.
```

```
IP=/sbin/ifconfig
IN=/sbin/ethtool
RT=/sbin/route
```

```
#Layer 2 configuration
```

```
$IN -s eth0 duplex full
$IN -s eth0 speed 100
```

```
$IN -s eth1 duplex full
$IN -s eth1 speed 100
```

```
$IN -s eth2 duplex full
$IN -s eth2 speed 100
```

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

```
$IN -s eth3 duplex full
$IN -s eth3 speed 100

#layer 3 Configuration

$IP eth1 3.0.100.12 netmask 255.255.255.248 up
$IP eth0 172.30.71.29 netmask 255.255.255.0 up
$IP eth2 172.30.40.4 netmask 255.255.255.0 up
$IP eth3 172.30.41.4 netmask 255.255.255.0 up

# Default Route

$RT add default gw 3.0.100.9

# Additional Routes

#end
```

Granted, this is a primitive example with no error checking, but it would configure all the needed interfaces, assuming the kernel loaded the drivers for them during boot. And it is this same approach, using the same commands, that the Linux boot scripts use to configure the server's network interface parameters during the boot process. The scripts that manage and configure the server's network interfaces are quite involved, but very flexible. Before we get into their mechanics, we need to review the Unix/Linux boot process.

The Unix/Linux systems employ a three-stage (OK, it's really two-stages, because the hardware BIOS is separate from the operating system load) bootstrap process. Starting from a power-off or "halted" state:

Stage 1:

BIOS: Probes for boot device.
BIOS: Loads the boot-loader.

Stage 2:

BOOT-LOADER: Loads the kernel into memory and hands over control.

Stage 3:

KERNEL: Identifies hardware and loads device drivers.

KERNEL: Initializes core processes (keventd, kswapd, bdflush) to schedule systems processes and manage system memory resources.

KERNEL: Initializes the run mode process.

There are seven Unix run modes (this is an approximation of the various Unix/Linux flavors):

- 0 - Halt
- 1 - Single user mode
- 2 - Multi-user, without networking
- 3 - Multi-user, with networking (normal/default mode)
- 4 - Unused, available for custom run mode

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

5 - X11, graphical login

6 - Reboot

The operational "run" level that the system "boots to" is set in the /etc/inittab configuration file. Each of the system "run levels" is achieved by the kernel loading a set of system configuration scripts that set the various run aspects of the server. The run-level scripts are located in the /etc/rc.d directory. Within /etc/rc.d are directories for each of the run levels rc0.d thru rc6.d, and within each of these directories are the actual load scripts (that are links to actual scripts located in /etc/init.d). Along with the run level script directories are the following:

- rc.sysinit sets all of the basic system attributes prior to loading the run level scripts, i.e., hostname, system clock, etc.
- rc starts and stops services during run-level changes.
- rc.local is the last run-level script executed. It provides a loader location for custom process initializations

Now, the reason for this exploration of the Linux boot process is the fact that the network configuration is part of the run-level initialization process. By default, the run-level script /etc/rc.d/rc3.d/S10network configures the server's network interfaces based on the configuration files located in /etc/sysconfig and /etc/sysconfig/network scripts. To be perfectly honest, there is an enormous amount of complexity here for bringing up a few router interfaces, but that is the price we pay for simplicity in the end. Let's dive in and take a look.

The simplest way to bring all of the interfaces up or down once the system is booted is to run the rc S10Network script:

```
/etc/rc3.d/S10network start
```

Here is the result:

```
[root@vulcan root]# /etc/rc3.d/S10network start
Setting network parameters:           [ OK ]
Bringing up loopback interface:       [ OK ]
Bringing up interface eth0:           [ OK ]
Bringing up interface eth1:           [ OK ]
```

The "stop" option will shut down the interfaces, so it's a good idea to only run this option from the console. The start option can be run remotely as long as you are not changing the IP address of the interface with which you have opened a remote VTY session. The basic behavior of S10network is determined by configuration values set in the /etc/sysconfig/network file, along with interface-specific option files located in the /etc/sysconfig/network-scripts directory. Alternatively, if you want bring up or bring down a specific host interface, the interface configuration scripts can be run from the command line using the ifup and ifdown control scripts located in /etc/sysconfig/network-scripts/. Here is the command syntax:

```
[root@vulcan network-scripts]# ifdown eth1
[root@vulcan network-scripts]# ifup eth1
```

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

The interface scripts provide no feedback. If you want to verify the status of the interface, you can use the `/sbin/ifconfig` command:

```
[root@vulcan network-scripts]# ifdown eth1

*****

[root@vulcan network-scripts]# ifconfig
eth0  Link encap:Ethernet HWaddr 00:30:48:43:86:98
      inet addr:172.30.71.22 Bcast:172.30.71.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1438 Metric:1
      RX packets:8603 errors:0 dropped:0 overruns:0 frame:0
      TX packets:419 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:1007070 (983.4 Kb) TX bytes:59923 (58.5 Kb)
      Base address:0xa000 Memory:ec000000-ec020000

[root@vulcan network-scripts]# ifup eth1

*****

[root@vulcan network-scripts]# ifconfig
eth0  Link encap:Ethernet HWaddr 00:30:48:43:86:98
      inet addr:172.30.71.22 Bcast:172.30.71.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1438 Metric:1
      RX packets:8636 errors:0 dropped:0 overruns:0 frame:0
      TX packets:439 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:1010277 (986.5 Kb) TX bytes:64027 (62.5 Kb)
      Base address:0xa000 Memory:ec000000-ec020000

eth1  Link encap:Ethernet HWaddr 00:30:48:43:86:99
      inet addr:172.30.40.4 Bcast:172.30.40.255 Mask:255.255.255.0
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:0 (0.0 b) TX bytes:704 (704.0 b)
      Base address:0xa400 Memory:ec020000-ec040000
```

Now let's take a look at the configuration scripts that manage the host's network interfaces. The place to start is with the configuration of `/etc/sysconfig/network`. Routers and switches use a single configuration file that's segmented into sections. Think of the `/etc/sysconfig/network` file as portion of the IOS configuration file that deals with the device-wide network parameters. The IP-related options for `/etc/sysconfig/network` can be as few as one and as many as six:

- `NETWORKING={yes|no}` – Enables/disables networking (required).
- `FORWARD_IPV4 = {yes|no}` – Enables/disables forwarding packets between server interfaces. This is one of three methods of enabling IP routing on Linux. This is optional; the default is no.
- `HOSTNAME = {name|name.subdomain.domain}` – Sets the server hostname value used by the server. If no value is set, the host defaults to "localhost." This value is reported with the `/bin/hostname` and `echo $HOSTNAME`. If you want the fully qualified hostname to be set, you

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

need to set the full hostname host.subdomain.domain. This is an optional value, but is really required for many applications to operate correctly.

- GATEWAY = {IP address of the default gateway in dotted-quad format *.*.*} – Sets the host's default IP default gateway. This is just one method for setting the DG. It is optional, but a DG needs to be set somehow for the server to have IP connectivity beyond its local subnets.
- GATEWAYDEV = "eth0" – Defines an interface rather than an IP address for the DG. This is optional and can be used in conjunction with the GATEWAY value or alone.

Sticking with our IOS configuration file analogy, there are three configuration files that deal with interface and routing parameters:

- /etc/sysconfig/network-scripts/ifcfg-eth* is the equivalent of an IOS interface stanza.
- /etc/sysconfig/network-scripts/route-eth* and /etc/sysconfig/static-routes are the static route definitions section.

An /etc/sysconfig/network-scripts/ifcfg-eth* file exists for each interface that the kernel manages during the boot process. Each configuration script has three required values and a number of relative options, depending on the required value settings:

DEVICE = Physical interface name – The LAN device name, i.e., eth0, fddi0, etc. Not optional.

ONBOOT = {yes|no} – Instructs the kernel to bring the interface up at boot. Not optional.

BOOTPROTO = {none|dhcp} – Instructs the kernel on how the interface gets its IP address information. Not optional.

NAME = Human name – Provides a human name reference for the interface. This is optional.

IPADDR = {IP address in dotted-quad format *.*.*} - Used with the BOOTPROTO = none option, it tells the kernel the interface's IP address.

NETMASK = {Subnet mask in dotted-quad format} - Used with the BOOTPROTO = none option, it tells the kernel the interface's subnet mask.

BROADCAST = {IP network's broadcast address in dotted-quad} – The network's broadcast address is the last IP address of the subnet allocation relative to the subnet mask. For example, the broadcast address for a host with an IP of 192.168.1.45 and subnet mask of 255.255.255.248 would be 192.168.1.47. This is optional and is deprecated and may be phased out.

NETWORK = {IP network's network address in dotted-quad} – The IP network's subnet ID or network address is the first IP address of the subnet allocation relative to the subnet mask. Again, a host with an IP address of 192.168.1.45/29 would have a subnet ID of 192.168.1.40. This is optional and is deprecated and may be phased out.

GATEWAY = {IP address of the default gateway in dotted-quad format *.*.*} - This is the second method for configuring the default route on a Linux host. This is optional and should be only set on one interface on multi-interface servers.

Standard Interface Configuration for a WLAN Proxy Server

Michael J. Martin

USERCTL = {yes|no} – Defines if a "standard" user can make configuration changes to the interface. This is optional; the default, however, is no user control.

MTU = {1-1500} – Sets the maximum transmission unit for the interface, defining the largest packet size the interface will transmit in a single frame. The default is 1500, which is fine for most applications. However, in applications where VLANs are used, the MTU should be reduced to accommodate the additional 4-byte frame payload used for the VLAN information tag. Although VLANs are supported by the Linux kernel, most Ethernet drivers do not support the additional overhead. There are some patches available for some drivers, but the lower MTU ensures there will not be any frame error problems. This is optional.

HWADDR = {ARP address in **:**:**:**:**:**} – Sets ARP address of the interface.

MRU = {1-1500} – Sets the maximum receive unit of the interface, defining the largest packet size the interface will accept. The same rule applies for VLAN interfaces, as with the MTU setting. Adjusting the MRU also is a crude way to broadly adjust the TCP maximum segment size (MSS) the host will transmit. Linux uses a default MSS value calculated from the first hop device MTU.

In situations where MTU discovery does not work because ICMP is blocked or disabled, adjusting the MSS configures the host to use smaller TCP packets, reducing the chances the packet will be fragmented. Fragmented packets are processed by router CPUs and are processed-switched. In network environments where IPsec and GRE are used, the additional payload requirements will often require large packets to be fragmented and impact router performance. This is optional. (Path-specific MSS can be set with the /sbin/route command).