CISH

Commandline Administrator's Guide

Pim van Riezen
Clue Consultancy

Table of Contents

Introduction....4
The Commandline....5
First Login....6
Configuring Ethernet....7
Configuring Static Routes....8
Configuring Serial Devices....10
Configuring Access Lists....12
Configuring IP Masquerading....14
The Syslog Service....15
The SNMP Service....16
The NTP Service....17
The Telnet Service....18
TCP/IP Tuning....19
Ethernet Bridging....20
Index....21

Introduction

CISH is a configuration shell for Linux−based network devices. It follows the *look and feel* of the configuration systems for large commercial routers. Using CISH as a primary interface to such a low cost device instead of the conventional UNIX working environment found on most Linux router systems offers a couple of nice advantages.

The UNIX shell is a very generic environment. In a traditional setting that is this shell's main advantage. However, in a more restricted context, like when configuring a networking appliance, a less generic interface brings tremendous usability benefits. In contrast with the UNIX shell, the CISH commandline is aware of the syntax of all its commands, allowing it to assist the operator with accurate online help and tab−completion. All that and no paperclip!

Because of the strong likeliness between CISH and the configuration shell of those larger commercial routers, the shell makes it easier to maintain a couple of low−cost appliances in the same space as expensive routing gear. This could be a positive thing for the level of acceptance of Linux appliances in the networking space.

CISH can be deployed on top of an installation of a Linux Router Project (LRP) system, functionally replacing most of the initialization and process management features of the regular distribution. The boot process for a CISH device is all derived from a single configuration file, which leads to a better overview of the entire system and a quicker startup with less respources going to management.

Using an LRP distribution with the Linux 2.2 kernel you can run an ethernet router with full support for IP masquerading, ethernet briding, extended access−lists, system logging, network time synchronization, remote configuration and ppp connectivity, quite nicely, from a 3.5" floppydisk on a machine with as little as 8MB of RAM.

This manual describes the commandline interface and the commandset available from the shell. In the *CISH Internals* document you can read about the implemtation in the UNIX environment and the organization of the source code.

We hope all this is useful to you and that you will gain experience. We also hope we will get an exclusive visit from the Swedish Bikini Team, so mind that we are not entirely discriminative on the subject of hope. We hope that is ok.

The Commandline

The Command Line Interface of CISH does not take much getting used to. In keybindings, it resembles the familiar interface of most popular UNIX shells as well as the configuration shell of commercial routers. When the *<TAB>* key is pressed, the current word under the cursor is completed to full length, unless if there are multiple possibilities. Pressing the question−mark *'?'* will show context−sensitive help about the available options.

The completion of commands with the *<TAB>* key makes it easier to grow proficiency in telling the interface what is needed as quickly and efficiently as possible. Note that if a command or argument gets completed by *<TAB>*, the shell does not actually need more information and a press en the space−bar or the *Enter* key will render the exact same interpretation as the hand−typed full command or the tab−completed equivalent.

The command prompt itself is subject to change during a typical configuration session if the user enters a *sub−shell* with a different command list. It is always possible to escape out of a sub−shell into the layer above it by issuing the ***done*** command. In contrast with other router shells, the *<CTRL><Z>* key combination is not in use as an escape out of a subshell. CISH universally uses the *<CTRL><D>* combination, which is more in par with a traditional UNIX commandline.

Keybindings Overview

Key or combination	***Function***
?	Get context−sensitive help
<TAB>	Complete the current keyword or list all the options.
<CTRL> <D>	Break out of a subshell
<CTRL> <C>	Break out of a foreground job
<CTRL> <A>	Jump to the beginning of the line
<CTRL> <E>	Jump to the end of the line
<CursUP>/<CursDOWN>	Scroll through the command history

First Login

When booted up for the first time, a system will generally not have a configured network device. The first login usually takes place on the (serial) console. When the first login prompt is shown, you are likely to run into this message:

```
Factory password presets configured
Please refer to documentation for login information
```

This means a factory configuration was loaded. The password for this initial login is the word "default". The first thing you should do is configure a new password: Log in and enter the privileged enable mode by typing the ***enable*** command. The system will ask for the second password, which is in the factory configuration also "default". Now issue the command to change the login and passwords:

```
router> enable
Password: (default)
router# secret login
Enter new password  : (new password)
Enter password again:
router# secret enable
Enter new password  : (new enable password)
Enter password again:
```

To save the new passwords you can use the ***write*** command, which will write the new configuration to the boot medium. Make sure you removedwrite protection mechanisms when doing so.

It may be a good idea to also assign a name to the router. Use the ***set hostname*** command from enable mode:

```
router# set hostname cable-masq
cable-masq#
```

Obviously this parameter doesn't affect anything outside the name the router presents on the prompt and when logging in. When you intend to use SNMP, read the chapter about the SNMP service on how to configure the owner, community and location parameters.

Configuring Ethernet

Now that the router is configured to no longer let the first malicious youth with a list of default passwords in, the time is ripe to configure the ethernet interfaces for tcp/ip. Let's assume that your router has two ethernet interfaces and will act as a gateway between them. The first segment has a network address of 10.0.0.0/24. The second segment has a network address of 192.168.0.0/23.

```
cable-masq# configure terminal
configure$ interface ethernet 0
interface% ip address 10.0.0.1 255.255.255.0
interface% no shutdown
interface% ^D
configure$ interface ethernet 1
interface% ip address 192.168.0.1 255.255.254.0
interface% no shutdown
interface% ^D
configure$ ^D
```

Your ethernet devices have now been configured. You can look at the status of the interfaces by using the ***show interfaces*** command:

```
cable-masq# show interfaces ethernet
Ethernet0 is up, line protocol is up
  Link protocol is Ethernet, hardware address is
0800.0980.2b64
  Internet address is 10.0.0.1 255.255.255.0
  Interface loopback is not set, MTU is 1500 bytes
  Interrupt 9, io-address 0x360
  Output queue size: 100
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 frame errors
     0 input errors, 0 dropped, 0 overruns
     0 packets output, 0 carrier errors
     0 output errors, 0 dropped, 0 overruns
     0 collisions detected

Ethernet1 is up, line protocol is up
  Link protocol is Ethernet, hardware address is
0800.09c0.371b
  Internet address is 192.168.0.1 255.255.254.0
  Interface loopback is not set, MTU is 1500 bytes
  Interrupt 11, io-address 0x380
  Output queue size: 100
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 frame errors
     0 input errors, 0 dropped, 0 overruns
     0 packets output, 0 carrier errors
     0 output errors, 0 dropped, 0 overruns
     0 collisions detected
```

Both interfaces seem to be up. You can check their configuration by using the ***ping*** command on a known host on either segment to verify that IP is working correctly now. You can also attempt to ping the router from another machine in one of the two segments.

Configuring Static Routes

Now that the router is capable of accepting and sending packets, it starts to become interesting to get it to do IP routing. First of all you have to make sure that IP forwarding has been disabled globally for the router:

```
cable-masq# configure terminal
configure$ ip forwarding
configure$
```

The interface configuration in the previous chapter already set up the segment routes for the two ethernet interfaces. You can check the routing table using the ***show ip route*** command.

```
cable-masq# show ip route
Network          Netmask          Route
10.0.0.0         255.255.255.0    Ethernet0
192.168.0.0      255.255.254.0    Ethernet1
127.0.0.0        255.0.0.0        Loopback0
```

Hosts in the 192.168.0.0/23 network, if they have their gateway address set to our interface on 192.168.0.1, can now reach the 10.0.0.0/24 network. And hosts on the latter can reah the 192.168.0.0/23 network if they have their default gateway configured to 10.0.0.1.

Let's assume that there is another router in your network which handles the network 10.0.1.0/24. It is connected to the 192.168.0.0 network on the address 192.168.1.1. Here's how you add a static route for this network:

```
cable-masq# configure terminal
configure$ ip route 10.0.1.0 255.255.255.0 192.168.1.1
configure$ ^D
cable-masq# show ip route
Network          Netmask          Route
10.0.0.0         255.255.255.0    Ethernet0
192.168.0.0      255.255.254.0    Ethernet1
127.0.0.0        255.0.0.0        Loopback0
10.0.1.0         255.255.255.0    192.168.1.1
```

The extra netblock is now routable. Machines in the 10.0.0.0 network can reach the 10.0.1.0 network through the 10.0.0.1 interface on Ethernet0.

Sometimes, multiple networks co-exist on the same broadcast domain. For example, in the segment behind Ethernet1 there may be another set of hosts operating within the 172.16.4.0/27 network:

```
cable-masq# configure terminal
configure$ ip route 172.16.4.0 255.255.255.224 Ethernet1
configure$ interface Ethernet1.0
interface% ip address 172.16.4.1 255.255.255.255
interface% no shutdown
interface% ^D
configure$ ^D
```

And you're all set. The hosts in the 172.16.4.0/27 range can talk to your router on the 172.16.4.1 alias interface we just brought up.

Lastly, you may want to have a *default* route defined for your router. Let's assume that there is a router on 192.168.1.254 that knows how to get packets to the outside world:

```
cable-masq# configure terminal
configure$ ip route 0.0.0.0 0.0.0.0 192.168.1.254
configure$ ^D
cable-masq# show ip route
Network          Netmask          Route
10.0.0.0         255.255.255.0    Ethernet0
192.168.0.0      255.255.254.0    Ethernet1
127.0.0.0        255.0.0.0        Loopback0
10.0.1.0         255.255.255.0    192.168.1.1
172.16.4.0       255.255.255.224  Ethernet1
0.0.0.0          0.0.0.0          192.168.1.254
```

Packets received for IP addresses your router receives for forwarding are now no longer its problem, they are sent to 192.168.1.254 to deal with.

Configuring Serial Devices

Serial devices connected to a CISH router can function either as login ttys or as outbound PPP connections. The available options when configuring a serial device are somewhat different from other (network−only) devices. To set up a serial device for a login console, apply the following magic:

```
cable−masq# configure terminal
configure$ interface serial 0
conf serial% speed 9600
conf serial% login
conf serial% flow−control rts−cts
conf serial% no shutdown
conf serial% ^D
configure$ ^D
cable−masq#
```

The first serial port is now configured for login with a terminal at 9600 baud and RTS/CTS flow−control. If you are using a three−wire serial connection you can also use XON/XOFF:

```
conf serial% flow−control xon−xoff
conf−serial% ^D
```

If the serial line is connected to another device and you want it to communicate through PPP, you have to set it up as ***ppp outbound*** instead of ***login*** as a type:

```
conf serial% speed 115200
conf serial% ppp outbound
conf serial% ip default−route
conf serial% ^D
```

This setup will leave the IP configuration of the connection to the peer machine. You can, of course, configure things explicitly as well:

```
conf serial% ip address 10.2.1.1 255.255.255.0
conf serial% ip peer−address 10.2.1.2
conf serial% ^D
```

Note that the configuration of the peer machine may be such that it will not accept our configuration but instead offer its own configuration. The factual result of the PPP connection can be determined from the ***show interfaces*** command.

```
cable−masq# show interfaces serial
Serial0 is up, line protocol is up
  Link protocol is Point−to−Point
  Internet address is 10.2.1.1 255.255.255.0
  Interface loopback is not set, MTU is 1500 bytes
  Output queue size: 10
  5 minute input rate 1838 bits/sec, 2 packets/sec
  5 minute output rate 1592 bits/sec, 2 packets/sec
     92897 packets input, 170 frame errors
     170 input errors, 0 dropped, 0 overruns
     85292 packets output, 0 carrier errors
     0 output errors, 0 dropped, 0 overruns
     0 collisions detected
```

There's a working PPP link brought up at the expected IP address. Note that, in contrast to the ppp−device on plain Linux systems, a network device configured under a numbered serial port like Serial0 is statically allocated. This behaviour is the same on most router systems.

Some PPP links need extra negociation with either the modem or the other machine to log in. This is where chat−scripts and the authentication options come in. Let's consider a dialout script which connects to the main PoP of Acme Dialin (212−555−1337):

```
cable-masq# configure terminal
configure$ chatscript 1 AT
configure$ chatscript 1 OK
configure$ chatscript 1 ATDT @dialout
configure$ chatscript 1 CONNECT
configure$ chatscript 1 ogin:
configure$ chatscript 1 @user
configure$ chatscript 1 ssword:
configure$ chatscript 1 @pass
configure$ interface serial0
conf serial% dialout 2125551337
conf serial% authentication chat 1
conf serial% authentication user wileec
conf serial% authentication pass b1rds33d
conf serial% ^D
```

The chat−script used here is very common among most dialin ISPs that use common RAS servers like the Lucent Portmaster.

Configuring Access Lists

Access Lists give the administrator a mechanism for keeping unwanted network packets outside certain network segments. CISH uses an administration system on top of the IP Chains technology in Linux which reflects the syntax and flexibility of commercial routers. Every interface on the router configured for networking can have an extended access−list assigned for inbound and/or outbound traffic on that port.

Let's assume an internet connection on the Ethernet0 device. On the other side of the router is a heterogenous network of workstations:

```
cable−masq# configure terminal
configure$ interface ethernet 0
interface% access−list 100 in
interface% ^D
configure$ access−list 100 deny tcp any any range 0−1023
configure$ access−list 100 deny udp any any range 0−1023
configure$ access−list 100 deny tcp any any eq 2049 syn
configure$ access−list 100 deny udp any any eq 2049
configure$ access−list 100 deny tcp any any eq 3128 syn
configure$ access−list 100 deny tcp any any eq 3306 syn
configure$ ^D
```

All traffic towards privileged ports is now blocked. Connection attempts (tcp packets with the SYN flag) towards certain extra ports (NFS, MySQL and a common proxy port) are also blocked. The rest is implicitly allowed.

Now let's say you want to make sure people on the inside network don't connect with the IllegalPornAndMp3Sharing.COM web− and ftp−sites to preserve bandwidth. These sites all reside on the 55.5.173.16/28 network. A simple ruleset:

```
cable−masq# configure terminal
configure$ access−list 100 deny ip 55.5.173.16 0.0.0.15
           any
configure$ ^D
```

You can remove the rules in an access−list using the ***no access−list*** command. CISH accepts netmask values in both forward and reverse notation and also allows you to remove a specific rule:

```
configure$ access−list 100 no deny tcp any any eq 2049
           syn
configure$ ^D
```

You can also insert a rule at the top of the access−list by prepending the ***insert*** keyword to a statement:

```
configure$ access−list 100 insert deny tcp any any eq
           31337
configure$ ^D
```

You can define up to 100 extended access−lists numbered from 100−199 and assign/unassign them freely to/from network interfaces. From 200 and upwards the global System access−lists are defined, representing the regular

Linux input, output and forward rule chains.

Using the ***show ip access−lists*** command you can take a look at the active configuration and the number of packets matched for each entry.

```
cable−masq# show ip access−lists
Extended IP access list 101
    permit ip 10.1.1.0 0.0.0.31 any  (6285 matches)
    deny   ip any any  (2 matches)
Extended IP access list 102
    deny   ip host 198.186.202.138 any  (390 matches)
    permit tcp any any eq 22  (50831 matches)
    permit tcp any any eq 113  (7330 matches)
    permit udp any eq 53 any eq 53  (0 matches)
    deny   tcp any any range 0−1023  (10101 matches)
    deny   tcp any host 10.0.1.2 eq 1026  (0 matches)
    deny   tcp any host 10.0.1.2 eq 1455  (0 matches)
    deny   tcp any host 10.0.1.2 eq 2048  (7 matches)
    deny   udp any any eq 2049  (0 matches)
    deny   tcp any any eq 2049  (26 matches)
    deny   tcp any any eq 3000  (829 matches)
    deny   tcp any any eq 3128  (18 matches)
    deny   tcp any any eq 3306  (745 matches)
    deny   tcp any any eq 4045  (548 matches)
    deny   udp any any eq 4045  (0 matches)
    deny   tcp any any eq 4321  (668 matches)
    deny   tcp any any eq 5232  (47 matches)
    deny   tcp any any eq 6000  (79 matches)
    deny   tcp any any eq 6112  (6 matches)
    deny   tcp any any eq 7100  (6 matches)
    deny   tcp any any eq 9100  (42 matches)
    deny   tcp any any eq 32771  (19 matches)
    deny   udp any any eq 32771  (0 matches)
    deny   tcp any any eq 32772  (5 matches)
    deny   tcp any any eq 32773  (8 matches)
    deny   tcp any any eq 32774  (5 matches)
    deny   tcp any any eq 32776  (19 matches)
    deny   ip 192.168.0.0 0.0.255.255 any  (38 matches)
    deny   ip 172.16.0.0 0.15.255.255 any  (103 matches)
    permit udp any eq 53 any eq 5353  (22592 matches)
    deny   ip 127.0.0.0 0.255.255.255 any  (0 matches)
cable−masq#
```

Configuring IP Masquerading

IP Masquerading is a many−to−one Network Address Translation scheme first implemented in the 1.2 kernel of Linux. It is popular for smaller gateway devices that interface with a dialup line or a cable modem. CISH supports this functionality through an acess−list and some extra parameters.

Let's assume that Ethernet0 is connected to a cable modem. Machines on the LAN are on Ethernet1 in the 192.168.0.0 range. A somewhat sensible setup would look like this:

```
cable−masq# configure terminal
configure$ access−list 201 masq ip 192.168.0.0 0.0.0.255
           any
configure$ ip masquerading tcp 7200
configure$ ip masquerading udp 300
configure$ ip masquerading fin 20
configure$ ^D
```

To really do a clean job, you make sure that no packets for the inner network which are not a part of the masquerading get routed.

```
cable−masq# configure terminal
configure$ interface ethernet 0
interface% access−list 103 in
interface% ^D
configure$ access−list 103 deny ip any 192.168.0.0
           0.0.0.255
configure$ access−list 103 deny ip 192.168.0.0 0.0.0.255
           any
configure$ ^D
```

The reason for the deny for packets destined to 192.168.0.0 seems counter−intuitive, until you remember that, before they enter the masquerading ruleset in access−list 201, the packets are still carry the *router* IP as their destination. Packets with that destination prior to the masquerading filter are therefor not kosher. The second rule prevents so−called *martian* source−addresses not to cross the interface. This is generally considered a good habit.

The Syslog Service

The Syslog facility is a well−known Unix service used to dispatch system events to a central source or a local logfile. CISH systems can log both to a remote resource and a local log which gets rotated automatically. The administrator can follow certain types of logging in real time from the command prompt. Let's look at how to configure local logging of certain access rules.

```
cable−masq# configure terminal
configure$ service syslog
configure$ log local
configure$ access−list 101 permit tcp any any eq 80 log
configure$ ^D
cable−masq# terminal debugging
cable−masq# debug access−lists
```

Logging to a remote facility is just as easy. Just specify the ip address of the server running a syslog daemon accepting network log events:

```
cable−masq# configure terminal
configure$ log remote 192.168.1.37
configure$ ^D
```

For instructions on how to set up a UNIX machine to receive syslog events, read the menual page for syslogd.

The SNMP Service

The Simple Network Management Protocol suite is commonly used by network hardware to communicate system status with external monitoring tools. CISH systems also have a non−complicated SNMP service. All that is needed is a couple of configuration lines:

```
cable-masq# configure terminal
configure$ service snmpd
configure$ snmp-server community jKaBBlaZek
configure$ snmp-server name cable-masq
configure$ snmp-server contact wile@acme.net
configure$ snmp-server location Grand Canyon
```

It is recommended to use a non−obvious community string. CISH routers only export read−only values in the current version, but the use of default or well−known community strings is generally seen as a dangerous thing.

The NTP Service

The Network Time Protocol is widely used to synchronize miscelaneous networked devices against a common clock. Configuration within CISH is quite straightforward, it entails enabling the NTP service and configuring an IP−address for the NTP master.

```
cable-masq# configure terminal
configure$ ntp-sync 60 192.168.1.14
```

With this configuration the router will synchronize every 60 seconds against the NTP server on 192.168.1.14.

The Telnet Service

Remote configuration can be a life saver for routing equipment. CISH allows the administrator to set up a remote configuration shell through the telnet service. Make note that telnet is an unencrypted protocol and therefor not suitable for transmission over an untrusted internet link. Enabling telnet only entails turning on the service:

```
cable-masq# configure terminal
configure$ service telnetd
```

If you want to prevent unauthorized access, it is desirable to use access-lists for denying packets aimed at the telnet port of any of the router interfaces.

TCP/IP Tuning

The Linux kernel exports a vast amount of tuning parameters to influence its behaviour in several ways. CISH allows the administrator to access these tuning parameters from configure mode through the ***ip*** command. The first family of tunable parameters influences the router's behaviour with regards to icmp packages destined to one of the router's own interfaces:

```
configure$ ip icmp rate echo-reply 1
configure$ ip icmp ignore broadcasts
```

There are a couple more like these. Check the commandline help to get an overview of most of the options.

A second family of tunable parameters deals with fragmenting and defragmenting of packets which pass the router. Refer to the Linux documentation for an in−depth discussion of this subject:

```
configure$ ip fragment always-defrag
```

More importantly, hidden in the ***ip*** family of commands are the settings for domainname and DNS nameserver:

```
configure$ ip domainname acme.net
configure$ ip nameserver 192.168.1.33
```

CISH tries to stay away from name−resolution as much as possible. The availability of a nameserver is merely a help to the operator in cases where it is convenient to ping or traceroute to a named host.

Ethernet Bridging

Since kernel 2.2, Linux has been capable of combining two or more physical ethernet segments into one using bridging. CISH allows you to configure specific ethernet devices to become part of a bridge group and use the Spanning Tree Protocol to prevent network loops:

```
cable-masq# configure terminal
configure$ interface ethernet 0
interface% bridge forwarding
interface% ^D
configure$ interface ethernet 1
interface% bridge forwarding
interface% ^D
configure$ bridge forwarding
configure$ bridge spanning-tree stp
```

There is also a "bridge" network device which you can use to configure a router interface that will listen on all ports that are part of the bridge forwarding group:

```
cable-masq# configure terminal
configure$ interface bridge 0
interface% ip address 192.168.1.0 255.255.255.0
interface% no shutdown
interface% ^D
configure$ ^D
```

You can determine whether an interface is part of a bridge group by taking a look at its output in the show command:

```
cable-masq# show interfaces ethernet 0
Ethernet0 is up, line protocol is up
  Link protocol is Ethernet, hardware address is
0050.0402.3be3
  Interface address is unnumbered
  Interface loopback is not set, MTU is 1500 bytes
  Interface is bridged, bridge port status is forwarding
  Interrupt 19, io-address 0xc400
  Output queue size: 100
  5 minute input rate 10917 bits/sec, 10 packets/sec
  5 minute output rate 23699 bits/sec, 9 packets/sec
     32751186 packets input, 8 frame errors
     7 input errors, 0 dropped, 0 overruns
     39465988 packets output, 0 carrier errors
     0 output errors, 0 dropped, 0 overruns
     0 collisions detected
```

The "bridge port status" field indicates how far the bridge group has advanced in its STP discovery phase. Normally it takes about 30 seconds for a bridge group to become fully active once it has been brought up using the ***bridge forwarding*** command.

Index

access−list....12
authentication chat....11
authentication pass....11
authentication user....11
bridge forwarding....20
bridge spanning−tree....20
bridging....20
chatscript....11
community....16
configure terminal....7
debug....15
dialout....11
DNS....19
enable....6
ethernet interfaces....7
factory configuration....6
Factory password....6
flow−control....10
help, context−sensitive....5
interface ethernet....7
interface serial....10
ip address....7, 10
ip domainname....19
ip forwarding....8
ip fragment....19
ip icmp ignore....19
ip icmp rate....19
ip masquerading....14
ip nameserver....19
ip peer−address....10
ip route....8
log local....15
log remote....15
LRP....4
martian....14
masq....14
nameserver....19
netmask....8
no access−list....12
NTP....17
ntp−sync....17
ppp outbound....10
secret enable....6
secret login....6
service snmpd....16
service syslog....15
service telnetd....18
set hostname....6

show interfaces.....7, 10
show ip access−lists.....13
show ip route.....8
SNMP.....16
snmp−server community.....16
snmp−server contact.....16
snmp−server location.....16
snmp−server name.....16
Spanning Tree Protocol.....20
speed.....10
STP.....20
sub−shell.....5
Swedish Bikini Team.....4
SYN.....12
syslog.....15
tab−completion.....5
telnet.....18
terminal debugging.....15
write.....6