The Linux Cipe+Masquerading mini−HOWTO

Table of Contents

Table of Contents

The Linux Cipe+Masquerading mini−HOWTO

Anthony Ciaravalo, acj@home.com

v1.2, 21 April 1999

How to setup a VPN using Cipe on a linux masquerading firewall.

---

1. Introduction

2. Firewall Configuration

3. Machine A Specific Configuration

4. Machine B Specific Configuration

5. Machine C Specific Configuration

6. Common Machine Configuration

7. Example masquerading firewall scripts

8. Putting it all together

9. Connecting to the WAN

10. References

---

1. Introduction

This is the Linux Cipe+Masquerading mini−HOWTO. It shows how to setup a Virtual Private Network between your LAN and other LAN's using Cipe on linux masquerading firewall machines. It also shows an example masquerading firewall configuration.

1.1 Copyright statement

If you have questions, please contact Tim Bynum, the Linux HOWTO coordinator, at tjbynum@wallybox.cei.net or linux−howto@metalab.unc.edu

1.2 Disclaimer

Use of the information and examples in this document is at your own risk. There are many security issues involved when connecting networks across the internet. Even though information is encrypted, an improperly configured firewall may result in a security breach. Precautions can be taken to protect your cipe connections, but it does not guarantee 100% security. The author does not guarantee the information provided in this document will provide a secure networking environment. Even though I have tried to be as accurate as possible creating this document, I am not responsible for any problems or damages incurred due to actions taken based on the information in this document.

1.3 Feedback

Send questions, comments, suggestions, or corrections to acj@home.com.

1.4 Getting the files

This howto was written based on Cipe versions 1.0.1 and 1.2.0. See reference section for link to Cipe home page.

---

2. Firewall Configuration

This howto assumes you already configured your kernel to support IP masquerade. See references below for information on configuring your kernel for a linux firewall.

2.1 VPN Network Diagram

This setup uses a star/hub configuration. It will set up a cipe connection from Machine A to Machine B and another from Machine A to Machine C.

```
                  Machine A
               eth0: 192.168.1.1
               eth1: real ip 1
              /               \
            /                  \
     Machine B                 Machine C
   eth0: 192.168.2.1          eth0:192.168.3.1
   eth1: real ip 2            eth1: real ip 3
```

2.2 A little reference

```
eth0 is the local network (fake address)
eth1 is the internet address (real address)

Port A is any valid port you would like to choose
Port B is any other valid port you would like to choose

Key A is any valid key you would like to choose  (read cipe doc for info)
Key B is any valid key you would like to choose
```

2.3 Additional notes about scripts and the VPN

The ip−up scripts currently only allow class c traffic through the cipe interface. If you wish for machine B to communicate with Machine C then you will need to change the appropriate ip−up and ip−down scripts.

Specifically, you need to change the ptpaddr and myaddr netmasks. There are two ip−up scripts, one for ipchains and one for ipfwadm. Same with the ip−down scripts. Change the appropriate incoming, outgoing, and forwarding cipe interface firewall rules netmask from /24 to /16. Any cipe firewall rule changes you make in ip−up for ipfwadm, make sure the ip−down script reflects the change so it will be properly removed from the list when the interface goes down. For the ipchains file, anything added in a chain does not need ip−down reflection since ip−down will flush all the rules in the user defined chain.

You will also need to uncomment the network route in the rc.cipe for Machine B and C that adds each others network to their route table.

3. Machine A Specific Configuration

3.1 /etc/cipe/options.machineB

```
#uncomment 1 below
#name for cipe 1.0.x
#device          cip3b0
#name for cipe 1.2.x
device          cipcb0

# remote internal (fake) ip address
ptpaddr         192.168.2.1
# my cipe (fake) ip address
ipaddr          192.168.1.1
# my real ip address and cipe port
me              (real ip 1):(port A)
# remote real ip address and cipe port
peer            (real ip 2):(port A)
#unique 128 bit key
key             (Key A)
```

3.2 /etc/cipe/options.machineC

```
#uncomment 1 below
#name for cipe 1.0.x
#device          cip3b1
#name for cipe 1.2.x
device          cipcb1

# remote internal (fake) ip address
ptpaddr         192.168.3.1
# my cipe (fake) ip address
ipaddr          192.168.1.1
# my real ip address and cipe port
me              (real ip 1):(port B)
# remote real ip address and cipe port
peer            (real ip 3):(port B)
#unique 128 bit key
key             (Key B)
```

3.3 /etc/rc.d/rc.cipe

```
!#/bin/bash
#rc.cipe  3/29/1999
#Send questions or comments to acj@home.com.

#Setup script path
PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#Options filenames in cipe directory for cipe interfaces
options="options.machineB options.machineC"

#Automatically obtain options filenames from cipe directory
#options=`/bin/ls /etc/cipe/options.*`

#Uncomment 1 below for the cipe module name
#cipemod="cip3b"        #for cipe 1.0
cipemod="cipcb"        #for cipe 1.2

#Check for cipe module and load if not already loaded
grep $cipemod /proc/modules >/dev/null
if [ "$?" = "1" ]; then
        echo Loading cipe module.
        modprobe $cipemod
        if [ "$?" = "1" ]; then
                echo Error loading cipe module...exiting.
                exit
        fi
else
        echo Cipe module already loaded.
fi

#Remove any existing cipe interfaces
cipeif=`cat /proc/net/dev | cut -f1 -d: | grep $cipemod`

if [ "$cipeif" != "" ]; then
        echo Removing existing cipe interface(s).
        for i in $cipeif; do
                ifconfig $i down
        done
fi

#Setup cipe interfaces
echo -n "Setting up cipe interface(s): "
for config in $options; do
        echo -n $config" "
        ciped -o $config
done
echo
echo

#Add routes for other remote networks via cipe interface(s)
#route add -net x.x.x.x netmask x.x.x.x gw x.x.x.x
```

3.4 Gateway

All machines on network 192.168.1.0 must have 192.168.1.1 as gateway. If you don't it will not work.

4. Machine B Specific Configuration

4.1 /etc/cipe/options.machineA

```
#uncomment 1 below
#name for cipe 1.0.x
#device          cip3b0
#name for cipe 1.2.x
device          cipcb0

#remote internal (fake) ip address
ptpaddr         192.168.1.1
# my cipe (fake) ip address
ipaddr          192.168.2.1
# my real ip address and cipe port
me              (real ip 1):(port A)
# remote real ip address and cipe port
peer            (real ip 2):(port A)
#unique 128 bit key
key             (Key A)
```

4.2 /etc/rc.d/rc.cipe

```
!#/bin/bash
#rc.cipe  3/29/1999
#Send questions or comments to acj@home.com.

#Setup script path
PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#Options filenames in cipe directory for cipe interfaces
options="options.machineA"

#Automatically obtain options filenames from cipe directory
#options=`/bin/ls /etc/cipe/options.*`

#Uncomment 1 below for the cipe module name
#cipemod="cip3b"        #for cipe 1.0
cipemod="cipcb"        #for cipe 1.2

#Check for cipe module and load if not already loaded
grep $cipemod /proc/modules >/dev/null
if [ "$?" = "1" ]; then
```

```
        echo Loading cipe module.
        modprobe $cipemod
        if [ "$?" = "1" ]; then
                echo Error loading cipe module...exiting.
                exit
        fi
else
        echo Cipe module already loaded.
fi

#Remove any existing cipe interfaces
cipeif=`cat /proc/net/dev | cut -f1 -d: | grep $cipemod`

if [ "$cipeif" != "" ]; then
        echo Removing existing cipe interface(s).
        for i in $cipeif; do
                ifconfig $i down
        done
fi

#Setup cipe interfaces
echo -n "Setting up cipe interface(s): "
for config in $options; do
        echo -n $config" "
        ciped -o $config
done
echo
echo

#Add routes for other remote networks via cipe interface(s)
#route add -net x.x.x.x netmask x.x.x.x gw x.x.x.x
#route to machine C network
#route add -net 192.168.3.0 netmask 255.255.255.0 gw 192.168.3.1
```

4.3 Gateway

All machines on network 192.168.2.0 must have 192.168.2.1 as gateway. If you don't it will not work.

5. Machine C Specific Configuration

5.1 /etc/cipe/options.machineA

```
#uncomment 1 below
#name for cipe 1.0.x
#device          cip3b0
#name for cipe 1.2.x
device          cipcb0

#remote internal (fake) ip address
ptpaddr         192.168.1.1
# my cipe (fake) ip address
ipaddr          192.168.3.1
# my real ip address and cipe port
```

```
me              (real ip 3):(port B)
#remote real ip address and cipe port
peer            (real ip 1):(port B)
#unique 128 bit key
key             (Key B)
```

5.2 /etc/rc.d/rc.cipe

```
!#/bin/bash
#rc.cipe  3/29/1999
#Send questions or comments to acj@home.com.

#Setup script path
PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#Options filenames in cipe directory for cipe interfaces
options="options.machineA"

#Automatically obtain options filenames from cipe directory
#options=`/bin/ls /etc/cipe/options.*`

#Uncomment 1 below for the cipe module name
#cipemod="cip3b"        #for cipe 1.0
cipemod="cipcb"        #for cipe 1.2

#Check for cipe module and load if not already loaded
grep $cipemod /proc/modules >/dev/null
if [ "$?" = "1" ]; then
        echo Loading cipe module.
        modprobe $cipemod
        if [ "$?" = "1" ]; then
                echo Error loading cipe module...exiting.
                exit
        fi
else
        echo Cipe module already loaded.
fi

#Remove any existing cipe interfaces
cipeif=`cat /proc/net/dev | cut -f1 -d: | grep $cipemod`

if [ "$cipeif" != "" ]; then
        echo Removing existing cipe interface(s).
        for i in $cipeif; do
                ifconfig $i down
        done
fi

#Setup cipe interfaces
echo -n "Setting up cipe interface(s): "
for config in $options; do
        echo -n $config" "
        ciped -o $config
done
echo
echo
```

```
#Add routes for other remote networks via cipe interface(s)
#route add -net x.x.x.x netmask x.x.x.x gw x.x.x.x
#route to machine B network
#route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.2.1
```

5.3 Gateway

All machines on network 192.168.2.0 must have 192.168.2.1 as gateway. If you don't it will not work.

6. Common Machine Configuration

6.1 /etc/cipe/ip−up

Kernel 2.0, ipfwadm, cipe 1.0.x

```
#!/bin/bash
# ip-up <interface> <myaddr> <daemon-pid> <local> <remote> <arg>
#3/29/1999
#An example ip-up script for the older 1.x 2.x kernels using ipfwadm that
#will setup routes and firewall rules to connect your local class c network
#to a remote class c network.

#The rules are configured to prevent spoofing and stuffed routing between
#the networks.  There are also additional security enhancements commented
#out towards the bottom of the script.
#Send questions or comments to acj@home.com.

#---------------------------------------------------------------------------
#Set some script variables
device=$1               # the CIPE interface
me=$2                   # our UDP address
pid=$3                  # the daemon's process ID
ipaddr=$4               # IP address of our CIPE device
vptpaddr=$5              # IP address of the remote CIPE device
option=$6               # argument supplied via options

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#comment/uncomment to enable/disbale kernel logging for all unauthorized
#access attempts. Must be same as ip-down script in order to remove rules.
log="-o"

#---------------------------------------------------------------------------
umask 022

# just a logging example
#echo "UP   $*" >> /var/adm/cipe.log

# many systems like these pid files
#echo $3 > /var/run/$device.pid

#---------------------------------------------------------------------------
```

```
#add route entry for remote cipe network
network=`expr $ptpaddr : '\([0-9]*\.[0-9]*\.[0-9]*\.\)'`0
route add -net $network netmask 255.255.255.0 dev $device

#need to add route entry for host in 2.0 kernels
route add -host $ptpaddr dev $device

#---------------------------------------------------------------------------
#cipe interface incoming firewall rules
#must be inserted into list in reverse order

#deny all other incoming packets to cipe interface
ipfwadm -I -i deny -W $device -S 0/0 -D 0/0 $log

#accept incoming packets from remotenet to localnet on cipe interface
ipfwadm -I -i accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#accept incoming packets from localnet to remotenet on cipe interface
ipfwadm -I -i accept -W $device -S $ipaddr/24 -D $ptpaddr/24

#deny incoming packets, cipe interface, claiming to be from localnet; log
ipfwadm -I -i deny -W $device -S $ipaddr/24 -D $ipaddr/24 $log

#---------------------------------------------------------------------------
#cipe interface outgoing firewall rules
#must be inserted into list in reverse order

#deny all other outgoing packets from cipe interface
ipfwadm -O -i deny -W $device -S 0/0 -D 0/0 $log

#accept outgoing from remotenet to localnet on cipe interface
ipfwadm -O -i accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#accept outgoing from localnet to remotenet on cipe interface
ipfwadm -O -i accept -W $device -S $ipaddr/24 -D $ptpaddr/24

#deny outgoing to localnet from localnet, cipe interface, deny; log
ipfwadm -O -i deny -W $device -S $ipaddr/24 -D $ipaddr/24 $log

#---------------------------------------------------------------------------
#The forwarding is configured so machines on your local network do not get
#masqueraded to the remote network.  This provides better access control
#between networks.  Must be inserted into list in reverse order

#deny all other forwarding through cipe interface; log
ipfwadm -F -i deny -W $device -S 0/0 -D 0/0 $log

#accept forwarding from remotenet to localnet on cipe interfaces
ipfwadm -F -i accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#accept forwarding from localnet to remotenet on cipe interfaces
ipfwadm -F -i accept -W $device -S $ipaddr/24 -D $ptpaddr/24

#---------------------------------------------------------------------------
#Make sure forwarding is enabled in the kernel. The kernel by default may
#have forwarding disabled.
/bin/echo 1 > /proc/sys/net/ipv4/ip_forward

#---------------------------------------------------------------------------
#Optional security enhancement - set default forward policy to
#DENY or REJECT.  If your forwarding default policy is DENY/REJECT
```

```
#you will need to add the following rules to your main forward chain.  It
#is a good idea to have all default policies set for DENY or
#REJECT.

#define machine interfaces
#localif="eth0"
#staticif="eth1"                ;cable modem users
#staticif="ppp0"                ;dialup users

#a real sloppy way to get the peer ip address from the options file - a new
#argument with peer ip:port passed to script would be nice.
#both lines need to be uncommented
#peerfile=`grep $device /etc/cipe/options.* | cut -f1 -d:`
#peer=`grep peer $peerfile | cut -f1 -d: | awk '{print $2}'`

#must log peer ip address for ip-down script
#echo $peer > /var/run/$device.peerip

#accept forwarding from localnet to remotenet on internal network interface
#ipfwadm -F -i accept -W $localif -S $ipaddr/24 -D $ptpaddr/24
#accept forwarding from remotenet to localnet on internal network interface
#ipfwadm -F -i accept -W $localif -S $ptpaddr/24 -D $ipaddr/24
#accept forwarding on staticif from me to peer
#myaddr=`echo $me | cut -f1 -d:`
#ipfwadm -F -i accept -W $staticif -S $myaddr -D $peer
#---------------------------------------------------------------------------
#Other optional security enhancement
#block all incoming requests from everywhere to our cipe udp port
#except our peer's udp port

#need to determine udp ports for the cipe interfaces
#get our udp port
#if [ "$option" = "" ]; then
#       myport=`echo $me | cut -f2 -d:`
#else
#       myport=$option
#fi

#get remote udp port -- peerfile variable must be set above
#peerport=`grep peer $peerfile | cut -f2 -d:`

#must log peer udp port for ip-down script
#echo $peerport > /var/run/$device.peerport

#get our ip address
#myaddr=`echo $me | cut -f1 -d:`

#deny and log all requests to cipe udp port must be inserted first
#ipfwadm -I -i deny -P udp -W $staticif -S 0/0 -D $myaddr $myport $log
#accept udp packets from peer at udp cipe port to my udp cipe port
#ipfwadm -I -i accept -P udp -W $staticif -S $peer $peerport \
#-D $myaddr $myport

exit 0
```

Kernel 2.1/2.2, ipchains, cipe 1.2.x

```
#!/bin/bash
# ip-up <interface> <myaddr> <daemon-pid> <local> <remote> <arg>
#3/29/1999
#An example ip-up script for the newer 2.1/2.2 kernels using ipchains that
#will setup routes and firewall rules to connect your local class c network
#to a remote class c network.  This script creates 3 user defined chains
#-input, output, and forward - for each cipe interface, based on the
#interface name. It will then insert a rule into each of the built-in
#input, output, and forward chains to use the user defined chains. The
#rules are configured to prevent spoofing and stuffed routing between the
#networks. There are also additional security enhancements commented out
#towards the bottom of the script.
#Send questions or comments to acj@home.com.

#---------------------------------------------------------------------------

#Set some script variables
device=$1               # the CIPE interface
me=$2                   # our UDP address
pid=$3                  # the daemon's process ID
ipaddr=$4               # IP address of our CIPE device
ptpaddr=$5              # IP address of the remote CIPE device
option=$6               # argument supplied via options

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#comment/uncomment to enable/disbale kernel logging for all unauthorized
#access attempts. Must be same as ip-down script in order to remove rules.
log="-l"

#---------------------------------------------------------------------------
umask 022
# just a logging example
#echo "UP   $*" >> /var/adm/cipe.log

# many systems like these pid files
#echo $3 > /var/run/$device.pid

#---------------------------------------------------------------------------
#add route entry for remote cipe network
network=`expr $ptpaddr : '\([0-9]*\.[0-9]*\.[0-9]*\.\)'`0
route add -net $network netmask 255.255.255.0 dev $device

#---------------------------------------------------------------------------
#create new ipchain for cipe interface input rules
ipchains -N $device"i"
#flush all rules in chain (sanity flush)
ipchains -F $device"i"
#deny incoming packets, cipe interface, claiming to be from localnet; log
ipchains -A $device"i" -j DENY -i $device -s $ipaddr/24 -d $ipaddr/24 $log
#accept incoming packets from localnet to remotenet on cipe interface
ipchains -A $device"i" -j ACCEPT -i $device -s $ipaddr/24 -d $ptpaddr/24
#accept incoming packets from remotenet to localnet on cipe interface
ipchains -A $device"i" -j ACCEPT -i $device -s $ptpaddr/24 -d $ipaddr/24
#deny all other incoming packets
ipchains -A $device"i" -j DENY -s 0/0 -d 0/0 $log

#---------------------------------------------------------------------------
#create new ipchain for cipe interface output rules
ipchains -N $device"o"
#flush all rules in chain (sanity flush)
ipchains -F $device"o"
```

```
#deny outgoing to localnet from localnet, cipe interface, deny; log
ipchains -A $device"o" -j DENY -i $device -s $ipaddr/24 -d $ipaddr/24 $log
#accept outgoing from localnet to remotenet on cipe interface
ipchains -A $device"o" -j ACCEPT -i $device -s $ipaddr/24 -d $ptpaddr/24
#accept outgoing from remotenet to localnet on cipe interface
ipchains -A $device"o" -j ACCEPT -i $device -s $ptpaddr/24 -d $ipaddr/24
#deny all other outgoing packets
ipchains -A $device"o" -j DENY -s 0/0 -d 0/0 $log

#---------------------------------------------------------------------------
#The forward chain is configured so machines on your local network do not
#get masqueraded to the remote network.  This provides better access
#control between networks.

#create new ipchain for cipe interface forward rules
ipchains -N $device"f"
#flush all rules in chain (sanity flush)
ipchains -F $device"f"
#accept forwarding from localnet to remotenet on cipe interfaces
ipchains -A $device"f" -j ACCEPT -i $device -s $ipaddr/24 -d $ptpaddr/24
#accept forwarding from remotenet to localnet on cipe interfaces
ipchains -A $device"f" -j ACCEPT -i $device -s $ptpaddr/24 -d $ipaddr/24
#deny all other forwarding; log
ipchains -A $device"f" -j DENY -s 0/0 -d 0/0 $log

#---------------------------------------------------------------------------
#Make sure forwarding is enabled in the kernel. New kernels by default have
#forwarding disabled.
/bin/echo 1 > /proc/sys/net/ipv4/ip_forward

#---------------------------------------------------------------------------
#insert rules to main input, output, and forward chains to enable new rules
#for the cipe interface
ipchains -I input -i $device -j $device"i"
ipchains -I output -i $device -j $device"o"
ipchains -I forward -i $device -j $device"f"

#---------------------------------------------------------------------------
#Optional security enhancement - set built-in forward chain policy to
#DENY or REJECT.  If your main forward chain default policy is DENY/REJECT
#you will need to add the following rules to your main forward chain.  It
#is a good idea to have all built-in chain default policies set for DENY or
#REJECT.

#define machine interfaces
#localif="eth0"
#staticif="eth1"                ;cable modem users
#staticif="ppp0"                ;dialup users

#a real sloppy way to get the peer ip address from the options file - a new
#argument with peer ip:port passed to script would be nice.
#both lines need to be uncommented
#peerfile=`grep $device /etc/cipe/options.* | cut -f1 -d:`
#peer=`grep peer $peerfile | cut -f1 -d: | awk '{print $2}'`

#must log peer ip address for ip-down script
#echo $peer > /var/run/$device.peerip

#accept forwarding from localnet to remotenet on internal network interface
#ipchains -I forward -j ACCEPT -i $localif -s $ipaddr/24 -d $ptpaddr/24
#accept forwarding from remotenet to localnet on internal network interface
#ipchains -I forward -j ACCEPT -i $localif -s $ptpaddr/24 -d $ipaddr/24
```

```
#accept forwarding on staticif from me to peer
#myaddr=`echo $me | cut -f1 -d:`
#ipchains -I forward -j ACCEPT -i $staticif -s $myaddr -d $peer
#---------------------------------------------------------------------------
#Other optional security enhancement
#block all incoming requests from everywhere to our cipe udp port
#except our peer's udp port

#need to determine udp ports for the cipe interfaces
#get our udp port
#if [ "$option" = "" ]; then
#       myport=`echo $me | cut -f2 -d:`
#else
#       myport=$option
#fi

#get remote udp port -- peerfile variable must be set above
#peerport=`grep peer $peerfile | cut -f2 -d:`

#must log peer udp port for ip-down script
#echo $peerport > /var/run/$device.peerport

#get our ip address
#myaddr=`echo $me | cut -f1 -d:`

#deny and log all requests to cipe udp port must be inserted first
#ipchains -I input -j DENY -p udp -i $staticif -s 0/0 \
#-d $myaddr $myport $log
#accept udp packets from peer at udp cipe port to my udp cipe port
#ipchains -I input -j ACCEPT -p udp -i $staticif -s $peer $peerport \
# -d $myaddr $myport

#---------------------------------------------------------------------------
# Set up spoofing protection in kernel as an additional security measure
#---------------------------------------------------------------------------
#Why do I have spoofing protection in the firewall rules in addition to
#this script that sets up spoof protection for each interface in the
#kernel? Guess I'm paranoid.

if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
        echo -n "Setting up IP spoofing protection..."
        iface="/proc/sys/net/ipv4/conf/$device/rp_filter"
        echo 1 > $iface
        echo "done."
else
        echo "Cannot setup spoof protection in kernel for $device" \
                | mail -s"Security Warning: $device" root
        exit 1
fi

exit 0
```

6.2 /etc/cipe/ip−down

Kernel 2.0, ipfwadm, cipe 1.0.x

```
#!/bin/bash

# ip-down <interface> <myaddr> <daemon-pid> <local> <remote> <arg>
#3/29/1999
#An example ip-down script for the older 1.x 2.x kernels using ipfwadm that
#will remove firewall rules that were setup to connect your local class c
#network to a remote class c network.

#---------------------------------------------------------------------------
#Set some script variables
device=$1               # the CIPE interface
me=$2                   # our UDP address
pid=$3                  # the daemon's process ID
ipaddr=$4               # IP address of our CIPE device
ptpaddr=$5              # IP address of the remote CIPE device
option=$6               # argument supplied via options

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#comment/uncomment to enable/disbale kernel logging for all unauthorized
#access attempts. Must be same as ip-down script in order to remove rules.
log="-o"

#---------------------------------------------------------------------------
umask 022

# just a logging example
#echo "DOWN   $*" >> /var/adm/cipe.log

# many systems like these pid files
#rm -f /var/run/$device.pid

#---------------------------------------------------------------------------
#cipe interface incoming firewall rules

#delete (deny all other incoming packets to cipe interface)
ipfwadm -I -d deny -W $device -S 0/0 -D 0/0 $log

#delete (accept incoming packets from remotenet to localnet on cipe
#interface)
ipfwadm -I -d accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#delete (accept incoming packets from localnet to remotenet on cipe
#interface)
ipfwadm -I -d accept -W $device -S $ipaddr/24 -D $ptpaddr/24

#delete (deny incoming packets, cipe interface, claiming to be from
#localnet and log)
ipfwadm -I -d deny -W $device -S $ipaddr/24 -D $ipaddr/24 $log

#---------------------------------------------------------------------------
#cipe interface incoming firewall rules

#delete (deny all other outgoing packets from cipe interface)
ipfwadm -O -d deny -W $device -S 0/0 -D 0/0 $log

#delete (accept outgoing from remotenet to localnet on cipe interface)
ipfwadm -O -d accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#delete (accept outgoing from localnet to remotenet on cipe interface)
ipfwadm -O -d accept -W $device -S $ipaddr/24 -D $ptpaddr/24
```

```
#delete (deny outgoing to localnet from localnet, cipe interface, deny
#and log)
ipfwadm -O -d deny -W $device -S $ipaddr/24 -D $ipaddr/24 $log

#---------------------------------------------------------------------------
#cipe interface forwarding firewall rules

#delete (deny all other forwarding through cipe interface; log)
ipfwadm -F -d deny -W $device -S 0/0 -D 0/0 $log

#delete (accept forwarding from remotenet to localnet on cipe interfaces)
ipfwadm -F -d accept -W $device -S $ptpaddr/24 -D $ipaddr/24

#delete (accept forwarding from localnet to remotenet on cipe interfaces)
ipfwadm -F -d accept -W $device -S $ipaddr/24 -D $ptpaddr/24

#---------------------------------------------------------------------------
#Optional security enhancement - set default forward policy to
#DENY or REJECT.  If your forwarding default policy is DENY/REJECT
#you will need to add the following rules to your main forward chain.  It
#is a good idea to have all default policies set for DENY or
#REJECT.

#define machine interfaces
#localif="eth0"
#staticif="eth1"                ;cable modem users
#staticif="ppp0"                ;dialup users

#a real sloppy way to get the peer ip address from the options file - a new
#argument with peer ip:port passed to script would be nice.
#both lines need to be uncommented
#peerfile=`grep $device /etc/cipe/options.* | cut -f1 -d:`
#peer=`grep peer $peerfile | cut -f1 -d: | awk '{print $2}'`

#must log peer ip address for ip-down script
#echo $peer > /var/run/$device.peerip

#delete (accept forwarding from localnet to remotenet on internal network
interface)
#ipfwadm -F -d accept -W $localif -S $ipaddr/24 -D $ptpaddr/24
#delete (accept forwarding from remotenet to localnet on internal network
interface)
#ipfwadm -F -d accept -W $localif -S $ptpaddr/24 -D $ipaddr/24
#delete (accept forwarding on staticif from me to peer)
#myaddr=`echo $me | cut -f1 -d:`
#ipfwadm -F -d accept -W $staticif -S $myaddr -D $peer
#---------------------------------------------------------------------------
#Other optional security enhancement
#block all incoming requests from everywhere to our cipe udp port
#except our peer's udp port

#need to determine udp ports for the cipe interfaces
#get our udp port
#if [ "$option" = "" ]; then
#       myport=`echo $me | cut -f2 -d:`
#else
#       myport=$option
#fi

#get remote udp port -- peerfile variable must be set above
#peerport=`grep peer $peerfile | cut -f2 -d:`
```

```
#must log peer udp port for ip-down script
#echo $peerport > /var/run/$device.peerport

#get our ip address
#myaddr=`echo $me | cut -f1 -d:`

#delete (deny and log all requests to cipe udp port must be inserted first)
#ipfwadm -I -d deny -P udp -W $staticif -S 0/0 -D $myaddr $myport $log
#delete (accept udp packets from peer at udp cipe port to my udp cipe port)
#ipfwadm -I -d accept -P udp -W $staticif -S $peer $peerport \
#-D $myaddr $myport

exit 0
```

Kernel 2.1/2.2, ipchains, cipe 1.2.x

```
#!/bin/sh
# ip-down <interface> <myaddr> <daemon-pid> <local> <remote> <arg>
#3/29/1999
#An example ip-down script for the newer 2.1/2.2 kernels using ipchains
#that will remove firewall rules that were setup to connect your local
#class c network to a remote class c network.  Optional security
#enhancement rules removal is also added and commented towards end of
#script.
#Send questions or comments to acj@home.com.

#---------------------------------------------------------------------------
#Set some script variables
device=$1               # the CIPE interface
me=$2                   # our UDP address
pid=$3                  # the daemon's process ID
ipaddr=$4               # IP address of our CIPE device
ptpaddr=$5              # IP address of the remote CIPE device
option=$6               # argument supplied via options
PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#comment/uncomment to enable/disbale kernel logging for all unauthorized
#access attempts
#must be same as ip-up script in order to remove rules
log="-l"

#---------------------------------------------------------------------------
umask 022

# Logging example
#echo "DOWN $*" >> /var/adm/cipe.log

# remove the daemon pid file
#rm -f /var/run/$device.pid

#---------------------------------------------------------------------------
#remove rules from main input, output, and forward chains for cipe
#interface
ipchains -D input -i $device -j $device"i"
ipchains -D output -i $device -j $device"o"
ipchains -D forward -i $device -j $device"f"
```

```
#-------------------------------------------------------------------------
#flush all rules in cipe interface input chain
ipchains -F $device"i"
#remove cipe interface input chain
ipchains -X $device"i"

#-------------------------------------------------------------------------
#flush all rules in cipe interface output chain
ipchains -F $device"o"
#remove cipe interface output chain
ipchains -X $device"o"

#-------------------------------------------------------------------------
#flush all rules in cipe interface forward chain
ipchains -F $device"f"
#remove cipe interface forward chain
ipchains -X $device"f"

#-------------------------------------------------------------------------
#Remove optional security enhancement rules

#get peer ip address
#peer=`cat /var/run/$device.peerip`

#define machine interfaces
#localif="eth0"
#staticif="eth1"                ;cable modem users
#staticif="ppp0"                ;dialup users

#get our ip address
#myaddr=`echo $me |cut -f1 -d:`

#delete (accept forwarding from localnet to remotenet on internal network
#interface)
#ipchains -D forward -j ACCEPT -i $localif -s $ipaddr/24 -d $ptpaddr/24
#delete (accept forwarding from remotenet to localnet on internal network
#interface)
#ipchains -D forward -j ACCEPT -i $localif -s $ptpaddr/24 -d $ipaddr/24
#delete (accept forwarding on staticif from me to peer)
#ipchains -D forward -j ACCEPT -i $staticif -s $myaddr -d $peer

#remove peer ip file
#rm /var/run/$device.peerip

#-------------------------------------------------------------------------
#Remove other optional security enhancement rules

#get peer udp port
#peerport=`cat /var/run/$device.peerport`

#get our udp port
#if [ "$option" = "" ]; then
#        myport=`echo $me | cut -f2 -d:`
#else
#        myport=$option
#fi

#delete (deny and log all requests to cipe udp port must be inserted first)
#ipchains -D input -j DENY -p udp -i $staticif -s 0/0 \
#-d $myaddr $myport $log
#delete (accept udp packets from peer at udp cipe port to my udp cipe port)
#ipchains -D input -j ACCEPT -p udp -i $staticif -s $peer $peerport \
```

```
#-d $myaddr $myport

#remove peer port file
#rm /var/run/$device.peerport

#----------------------------------------------------------------------------

exit 0
```

7. Example masquerading firewall scripts

7.1 Kernel 2.0, ipfwadm

```
#!/bin/sh
#04/04/1999
#example rc.firewall script for the 2.0 kernels using ipfwadm
#I cant take full credit for this script.  I had found it a few
#years ago and made slight modifications.
#Send questions or comments to acj@home.com.

#----------------------------------------------------------------------
#Variables
#----------------------------------------------------------------------

#local ethernet interface
localip=
localif=eth0

#static ethernet interface
staticip=
staticif=eth1

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#----------------------------------------------------------------------
#Incoming Firewall Policies
#----------------------------------------------------------------------
#flush incoming firewall policies
/sbin/ipfwadm -I -f

#set incoming firewall policy default to deny
/sbin/ipfwadm -I -p deny

#----------------------------------------------------------------------

#local interface, local machines, going anywhere is valid
/sbin/ipfwadm -I -a accept -V $localip -S $localip/24 -D 0.0.0.0/0
#remote interface, claiming to be local machines (IP spoofing) deny and log
/sbin/ipfwadm -I -a deny -V $staticip -S $localip/24 -D 0.0.0.0/0 -o
#remote interface, any source, going to staticip address is valid
/sbin/ipfwadm -I -a accept -V $staticip -S 0.0.0.0/0 -D $staticip/32
#loopback interface is valid
/sbin/ipfwadm -I -a accept -V 127.0.0.1 -S 0.0.0.0/0 -D 0.0.0.0/0
#all other incoming is denied and logged
```

```
/sbin/ipfwadm -I -a deny -S 0.0.0.0/0 -D 0.0.0.0/0 -o

#-----------------------------------------------------------------------
#Outgoing Firewall Policies
#-----------------------------------------------------------------------

#flush outgoing firewall policies
/sbin/ipfwadm -O -f

#set outgoing firewall policy default to deny
/sbin/ipfwadm -O -p deny

#-----------------------------------------------------------------------

#local interface, any source going to local net is valid
/sbin/ipfwadm -O -a accept -V $localip -S 0.0.0.0/0 -D $localip/24
#outgoing to localnet on static interface, stuffed routing, deny
/sbin/ipfwadm -O -a deny -V $staticip -S 0.0.0.0/0 -D $localip/24 -o
#outgoing from localnet on static interface, stuffed masquerading, deny
/sbin/ipfwadm -O -a deny -V $staticip -S $localip/24 -D 0.0.0.0/0 -o
#outgoing to localnet on static interface, stuffed masquerading, deny
/sbin/ipfwadm -O -a deny -V $staticip -S 0.0.0.0/0 -D $localip/24 -o
#anything else outgoing on remote interface is valid
/sbin/ipfwadm -O -a accept -V $staticip -S $staticip/32 -D 0.0.0.0/0
#loopback interface is valid
/sbin/ipfwadm -O -a accept -V 127.0.0.1 -S 0.0.0.0/0 -D 0.0.0.0/0
#all other outgoing is denied and logged
/sbin/ipfwadm -O -a deny -S 0.0.0.0/0 -D 0.0.0.0/0 -o

#----------------------------------------------------------------------------
#Forwarding firewall policies
#----------------------------------------------------------------------------

#flush forwarding policies
/sbin/ipfwadm -F -f

#set forwarding policy default to deny
/sbin/ipfwadm -F -p deny

#masquerade from localnet on local interface to anywhere
/sbin/ipfwadm -F -a masquerade -W $staticif -S $localip/24 -D 0.0.0.0/0
#all other forwarding is denied
/sbin/ipfwadm -F -a deny -S 0.0.0.0/0 -D 0.0.0.0/0

exit 0
```

7.2 Kernel 2.1/2.2, ipchains

```
#!/bin/sh
#04/04/1999
#example rc.firewall script for the newer 2.1/2.2 kernels using ipchains
#that creates user defined chains for each interface.  There are firewall
#rules for spoofing protection which may be unnecessary since the newer
#kernels can have kernel spoofing protection enabled.  You might say it's
#super paranoid checking.
#Send questions or comments to acj@home.com.
```

```
#----------------------------------------------------------------------
#Variables
#----------------------------------------------------------------------

#local ethernet interface
localip=
localif=eth0

#static ethernet interface
staticip=
staticif=eth1

#loopback interface
loopback=lo

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

#----------------------------------------------------------------------
#Flush built-in input, output, and forward ipchains; set default policy
#Good policy to deny all packets especially while setting up chains
#----------------------------------------------------------------------

#set incoming firewall policy default to deny
ipchains -P input DENY

#flush incoming firewall policies
ipchains -F input

#----------------------------------------------------------------------

#set outgoing firewall policy default to deny
ipchains -P output DENY

#flush outgoing firewall policies
ipchains -F output

#----------------------------------------------------------------------

#set forwarding firewall policy default to deny
ipchains -P forward DENY

#flush forwarding firewall policies
ipchains -F forward

#----------------------------------------------------------------------
#flush all policies  -redundant for main policies, but also flushes user
#defined policies
#ipchains -F

#remove all user defined policies - you may or may not want to enable this
#ipchains -X

#----------------------------------------------------------------------
#Incoming Firewall Policies
#----------------------------------------------------------------------

#create new input chain for static ethernet interface
ipchains -N $staticif"-i"

#flush all rules in chain (sanity flush)
ipchains -F $staticif"-i"
```

```
#block incoming tcp SYN packets to all ports on staticif and log
#this may be a little harsh but its a nice feature
#ipchains -A $staticif"-i" -j DENY -p tcp -y -i $staticif -s 0/0 \
#-d $staticip : -l

#remote interface, claiming to be local machines (IP spoofing) deny and log
ipchains -A $staticif"-i" -j DENY -i $staticif -s $localip/16 -d 0/0 -l

#remote interface, any source, going to staticip address is valid
ipchains -A $staticif"-i" -j ACCEPT -i $staticif -s 0/0 -d $staticip/32

#all other incoming is denied and logged
ipchains -A $staticif"-i" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new input chain for local ethernet interface
ipchains -N $localif"-i"

#flush all rules in chain (sanity flush)
ipchains -F $localif"-i"

#local interface, local machines, going anywhere is valid
ipchains -A $localif"-i" -j ACCEPT -i $localif -s $localip/24 -d 0/0

#all other incoming is denied and logged
ipchains -A $localif"-i" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new input chain for loopback interface
ipchains -N $loopback"-i"

#flush all rules in chain (sanity flush)
ipchains -F $loopback"-i"

#loopback interface is valid
ipchains -A $loopback"-i" -j ACCEPT -i $loopback -s 0/0 -d 0/0

#all other incoming is denied and logged
ipchains -A $loopback"-i" -j DENY -s 0/0 -d 0/0 -l

#---------------------------------------------------------------------------
#Forwarding firewall policies
#---------------------------------------------------------------------------

#create new forward chain for static ethernet interface
ipchains -N $staticif"-f"

#flush all rules in chain (sanity flush)
ipchains -F $staticif"-f"

#masquerade from localnet on static interface to anywhere
ipchains -A $staticif"-f" -j MASQ -i $staticif -s $localip/24 -d 0/0

#all other forwarding is denied and logged
ipchains -A $staticif"-f" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new forward chain for local ethernet interface
ipchains -N $localif"-f"
```

```
#flush all rules in chain (sanity flush)
ipchains -F $localif"-f"

#all other forwarding is denied and logged
ipchains -A $localif"-f" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new forward chain for loopback interface
ipchains -N $loopback"-f"

#flush all rules in chain (sanity flush)
ipchains -F $loopback"-f"

#all other forwarding is denied and logged
ipchains -A $loopback"-f" -j DENY -s 0/0 -d 0/0 -l


#----------------------------------------------------------------------
#Outgoing Firewall Policies
#----------------------------------------------------------------------

#create new output chain for static ethernet interface
ipchains -N $staticif"-o"

#flush all rules in chain (sanity flush)
ipchains -F $staticif"-o"

#outgoing to localnet on remote interface(stuffed routing) deny & log
ipchains -A $staticif"-o" -j DENY -i $staticif -s 0/0 -d $localip/24 -l

#outgoing from local net on remote interface, stuffed masquerading, deny
ipchains -A $staticif"-o" -j DENY -i $staticif -s $localip/24 -d 0/0 -l

#anything else outgoing on remote interface is valid
ipchains -A $staticif"-o" -j ACCEPT -i $staticif -s $staticip/32 -d 0/0

#all other outgoing is denied and logged
ipchains -A $staticif"-o" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new output chain for local ethernet interface
ipchains -N $localif"-o"

#flush all rules in chain (sanity flush)
ipchains -F $localif"-o"

#local interface, any source going to local net is valid
ipchains -A $localif"-o" -j ACCEPT -i $localif -s 0/0 -d $localip/24

#all other outgoing is denied and logged
ipchains -A $localif"-o" -j DENY -s 0/0 -d 0/0 -l

#----------------------------------------------------------------------

#create new output chain for loopback interface
ipchains -N $loopback"-o"

#flush all rules in chain (sanity flush)
ipchains -F $loopback"-o"
```

```
#loopback interface is valid
ipchains -A $loopback"-o" -j ACCEPT -i $loopback -s 0/0 -d 0/0
#all other outgoing is denied and logged
ipchains -A $loopback"-o" -j DENY -s 0/0 -d 0/0 -l

#---------------------------------------------------------------------------
#make sure forwarding is enabled in the kernel
#---------------------------------------------------------------------------

/bin/echo 1 > /proc/sys/net/ipv4/ip_forward

#---------------------------------------------------------------------------
#Add pointers to built-in chains to enable user defined chains
#change the order in each chain to optimize filtering for an interface
#---------------------------------------------------------------------------

#add local interface input chain
ipchains -A input -i $localif -j $localif"-i"

#add static interface input chain
ipchains -A input -i $staticif -j $staticif"-i"

#add loopback interface input chain
ipchains -A input -i $loopback -j $loopback"-i"

#--------------------------------------------------------------------------

#add local interface output chain
ipchains -A output -i $localif -j $localif"-o"

#add static interface output chain
ipchains -A output -i $staticif -j $staticif"-o"

#add loopback interface output chain
ipchains -A output -i $loopback -j $loopback"-o"

#--------------------------------------------------------------------------

#add local interface forward chain
ipchains -A forward -i $localif -j $localif"-f"

#add static interface forward chain
ipchains -A forward -i $staticif -j $staticif"-f"

#add loopback interface forward chain
ipchains -A forward -i $loopback -j $loopback"-f"

#----------------------------------------------------------------------
#Super Paranoid check --- even though default policy is set for deny,
#block all packets on any interface
#----------------------------------------------------------------------

#all other incoming is denied and logged
ipchains -A input -j DENY -s 0/0 -d 0/0 -l

#all other output is denied and logged
ipchains -A output -j DENY -s 0/0 -d 0/0 -l

#all other forwarding is denied and logged
ipchains -A forward -j DENY -s 0/0 -d 0/0 -l
```

```
exit 0
```

8. Putting it all together

This is an example rc.local script to start everything when your system boots. It will add spoofing protection in the kernel if you are using a 2.2 kernel, setup the masquerading firewall policies, and start the cipe interface(s).

```
#!/bin/bash
#4/4/99
#an example rc.local script
#Send questions or comments to acj@home.com

echo

#Set up spoof protection in kernel -- from IPChains HOWTO by Paul Russell

#this is only for the newer 2.1/2.2 kernels

#if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
#   echo -n "Setting up IP spoofing protection..."
#   for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
#       echo 1 > $f
#   done
#   echo "done."
#else
#   echo PROBLEMS SETTING UP IP SPOOFING PROTECTION.  BE WORRIED.
#   echo "CONTROL-D will exit from this shell and continue system startup."
#   echo
#   # Start a single user shell on the console
#   /sbin/sulogin $CONSOLE
#fi

echo

#Setup firewall policies
if [ -x /etc/rc.d/rc.firewall ]; then
        echo Setting up firewall packet filtering policies.
        echo
        . /etc/rc.d/rc.firewall
fi

#Start cipe interfaces
if [ -x /etc/rc.d/rc.cipe ]; then
        echo Starting VPN interfaces.
        . /etc/rc.d/rc.cipe
fi

exit 0
```

9. Connecting to the WAN

At this point your cipe interface should be up and running. Try pinging machines on the other network(s). If you cannot ping check the following on the firewall machine:

- Check that forwarding is enabled in the kernel.
- Do an ifconfig to check if the cipe interface is up.

```
cipcb0  Link encap:IPIP Tunnel  HWaddr
        inet addr:192.168.1.1  P-t-P:192.168.2.1  Mask:255.255.255.255
        UP POINTOPOINT NOTRAILERS RUNNING NOARP  MTU:1442  Metric:1
        RX packets:28163 errors:6 dropped:0 overruns:0 frame:6
        TX packets:29325 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:100
```

- Check the route table for a host entry for the other cipe host on the cipe interface.

```
192.168.2.1    *      255.255.255.255 UH    0   0        0 cipcb0
```

- Check the route table for a network entry to the other network(s) on the cipe interface.

```
192.168.2.0    *      255.255.255.0   U     0   0        0  cipcb0
```

- Check the log files for any error messages.

If your other machines behind your firewall cannot access machines behind the other firewall check that the gateway is properly setup on both ends.

Once you are able to ping, ftp, telnet, etc. to machines on the other network, the next step is to get your networks to see each other and access each other using SAMBA browsing. A few hints: lmhosts or wins server is required, trusted domains for NT. I have set these up, but that is not the purpose of this document (at least not for now).

If you used the example firewall masquerading script, then all of your machines should also be able to connect to the internet. If you cannot, then you might want to check the log files. You may also want to try using tcpdump to see what is happening with the packets.

---

10. References

10.1 Web Sites

Cipe Home Page

Masq Home Page

Samba Home Page

Linux HQ −−−great site for lots of linux info

10.2 Documentation

cipe.info: info file included with cipe distribution

Firewall HOWTO, by Mark Grennan, markg@netplus.net

IP Masquerade mini−HOWTO,by Ambrose Au, ambrose@writeme.com

IPChains−Howto, by Paul Russell, Paul.Russell@rustcorp.com.au

---