Installing OpenWrt
“White “Russian 0.9”
on a
Linksys WRT54GSv1.1

Author: Chris Martin
Email: chris@martin.name

Document Version: 1.8

Last updated: 1st March 2007

Overview

This document now assumes that OpenWrt “White Russian 0.9” is the version to be installed, that the target platform is a WRT54GS(v1.1) and that the OpenWRT distribution is using the squash file-system (earlier documents based on RC2, RC4, and RC5 used the JFFS file system, however this now strongly discouraged by the OpenWRT developers, and I am following suit, especially now that the mini_fo module is available as it minimises some of the inconvenience of the squash fs.).

Conventions

In this document the following conventions have been used

Scripts that should be “cut & paste” in to a terminal emulation session on the Linksys are in a courier font and surrounded with a border. These scripts have some characters escaped to allow then to them to be pasted to the command line. If you plan to edit the files manually, then you will need to remove the leading “\” that is used to escape special characters eg

Highlighted lines are the lines that expect that you will need to customise/modify for your configuration

```
# This is a sample script
echo “your name is: “\c

#This line may need to be customised
echo “Mike Smith”

# bye
exit 0
```

If there is no border around the text then it is a sample file and you will need to manually edit the file: eg

```
#!/bin/sh
# This is a sample script
echo “your name is: “\c

#This line may need to be customised
echo “Mike Smith”

# bye
exit 0
```

Where ever possible I have attempted to place all configuration files in the /etc directory to make it easier to back them all up.
The scripts that I have used I have placed in /etc/bin, for the same reason

Installation

Download the latest image from:
http://downloads.openwrt.org/whiterussian/0.9/

make sure you read the readme to get the build that supports the connectivity options you require. I am using the pppoe version. You may require a one of the others

each build is also bundled in two formats, *.bin, and *.trx. The *.bin is identical to the *.trx, with the exception that it includes a header identifying the target device.

JFFS file system images are no longer downloadable from OpenWrt, and the use of the JFFS file system is discouraged. If you really want to use a JFFS file system you will need to download the build system and build your own image.

The rest of this document will assume the use of the squashfs file system.

NOTE: There are two versions of the firmware for the wrt54gs.

- If you are using a wrt54gs prior to version 4, use the standard wrt54gs bin file
- openwrt-wrt54gs-squashfs.bin.
- If you are using a wrt54gs version 4 then obviously use the image specifically for the wrt54gsv4.
- openwrt-wrt54gs_v4-squashfs.bin
- If you have version 5. I'm sorry; but OpenWrt will not ever run on this device

NVRAM parameters (and therefore the majority of the router configuration), should be preserved across reboots. This means that your current IP address and other configuration will persist after the update. However, just in case this doesn't occur you will need to know the default network configuration for the router.
For the Linksys wrt54gs the following are the defaults:

	IP Address	**Subnet Mask**
router	192.168.1.1	255.255.255.0
PC workstation	dynamically assigned by DHCP from the router In the event that it is not provided, manually set the IP address to 192.168.1.2	255.255.255.0
dns server	192.168.1.1	N/A
gateway	192.168.1.1	N/A

PS. Make sure you connect your PC to one of the LAN ports and not the Internet port

Installing over the Linksys firmware

If the router still has the Linksys firmware loaded, then upload the new "*.bin" image via the firmware upload webpage.

Installing over previous openWrt firmware

If the router already has the openWrt firmware loaded then the easiest method is to tftp another image to the router. Make sure that the initial NVRAM parameters are set (in particular the “boot_wait” parameter should be set to on). See the “initial NVRAM variables” section below

NOTE 1: nvram parameters should be preserved across updates. Now ever it would be a wise move to make a back up of the following just in case up need to reference them in the future

1. *All nvram settings (nvram show > /tmp/nvramSettings.txt)*
2. *All installed packages (ipkg list_installed > /tmp/ipkgList.txt)*
3. *The /etc directory (cd /etc; tar –cvf /tmp/etc.tar .)*

Then copy the files to off the router. I have found the easiest way to do this is to use the scp command and copy them to a Linux machine. If you don’t have access to a Linux machine then, you can copy or link the files to the openWrt web server and access them from there.

NOTE 2: When re-flashing the router. The NVRAM “boot_wait parameter causes the router to delay loading from its firmware, as a result, it will not have loaded the parameters from NVRAM, and will be using its default IP address. When re-flashing the router you must access it via the default address.

To re-flash the router:

1. Plug a network cable between the switch and your PC/laptop. You MUST connect to one of the router LAN ports. DO NOT connect to the Internet port. Also you CAN NOT flash the over the wireless network.
2. Set the IP address of you workstation to and address in the 192.168.1.0/24 subnet. Best to use 192.168.1.2 if it is available. If you need to it is OK to set multiple IP address on your network interface so that you can still connect to the router using its configured IP address.
3. Logon to the router and issue the commands
   ```
   nvram set boot_wait=on
   nvram commit
   ```
4. Remove the power from the router
5. Start the tftp transfer with the router off
   a. From a windows PC issue the command:
      ```
      tftp -i 192.168.1.1 put image.bin
      ```
   b. If you are using a linux PC use the command. You may have trouble using the std linux tftp utility. Install the “Advanced tftp” (atftp) package
6. Then immediately power-up the router
7. Once the router has been re-flashed, the power light will flash for a while. This is the period that the file system is being rebuilt/updated. When the power light shops flashing and the DMZ light goes out, the update has completed, and you can telnet to the router. However the file-system is in a read-only state, and the router needs to be rebooted again so that the root file-system can be mounted read/write
8. Reboot the router
9. Once the router has been rebooted, you can then login/telnet to the router, and use the passwd command to set a password, once the password has been set the telnet service will be blocked and you will need to use SSH next time that you login.

Note: Some interesting information of the use of the LEDs.

LED	Meaning
POWER Flashing *(goes on and off at intervals of 1 second or less)*	*The power led can flash for a number of reasons; a flashing power led by itself is rarely cause for concern. The power led will flash on bootup until the firmware stops the flashing, but it may start flashing again with certain wireless settings (eg: configured as wifi client, not connected)*
DMZ	*OpenWrt uses the DMZ led to signal bootup. Once the kernel has booted OpenWrt will turn on the DMZ led; it won't turn off again until after OpenWrt has processed all the startup scripts in /etc/init.d. If it doesn't turn off then there is a problem with one of the startup scripts.*
DMZ flashing *(quickly flashes 3 times a second)*	*The router has successfully entered FailSafe mode*
POWER flashing DMZ alternating	*Your router is in a reboot loop, probably because you flashed a **corrupt** image.* *This might be because you did not set the binary flag on your tftp program, or have a bad image, or your upload was interrupted.* *Just reflash, and make sure your tftp program is configured correctly (ie: binary). And check you have a clean and correct image for your router.*

Erasing NVRAM variables

Warning … Warning … Warning … Warning ... Warning ... Warning ... Warning

Only do this on a WRT54GSV1.1 or other NVRAM save router

Not all routers using NVRAN are save to erase. Some models are unable to regenerate the required NVRAM settings and unit may be bricked. The only way to recover is to get an copy of the NVRAM from another identical router and load them using JTAG

If you want to re-configure your router form scratch, you will need to remove all the configuration from the NVRAM.

If you are installing OpenWrt for the first time, it makes things a bit easier if to you erase the NVRAM to remove redundant NVRAM settings left over from the original Linksys firmware. (If you are upgrading from a previous OpenWrt version, I would leave the NVRAM settings as the configuration will persist across the upgrade). There are a large number of parameters that are defined by the Linksys firmware that you will not use. If you erase them openWrt, once the router has been rebooted, it will recreate the ones it needs with default settings. A very good place to start from!.

To erase nvram, ie: reset to defaults, do the following

```
# Clear nvram
mtd erase nvram

#restart
Reboot
```

This will have the same effect as performing a hard reset. Ie: power on, with the reset button pressed.

Setting recovery NVRAM variables

These are the nvram variables that should always be set. When messing with new images, it is a good idea to make sure that these two are always set

NOTE: *only set the clkfreq parameter if you are using a linksys54gs v1.1*
This parameter sets the clock frequency from its default of 200Mhz to 216Mhz. This is required to add stability. Many people reported that their routers failed under high loads. Linksys also had the same problem, and modified this parameter to resolve a timing issue that was the cause.

NOTE2: *Setting the clkfreq may now be obsolete, It appears that the openwrt guys detect the hardware and set the parameter at boot time, or have found another way to solve the timing issue.*

```
nvram set boot_wait=on  # enable a delay to allow tftp upload on boot
#nvram set clkfreq=216  # slightly over clock the processor,
                        # fixes a stability problem
nvram commit            # commit changes from memory to NVRAM
```

Note: If you hard reset the router, all parameters will be reset back to defaults.
In particular this means that you will lose the “boot_wait” setting, which may make recovery of the router after a bad firmware upgrade much more difficult. Best to set it after each hard reset, or add it to the start-up script on the router

Recovering when things go wrong

It is possible (but very rare) to create a configuration that causes the router to lock-up. This is refered to as “bricking” your router (ie: it is now a brick). Fortunately the openWrt developers provided us with two ways to recover the router, bootwait and failsafe. If these don’t get you out of trouble, two more methods are available, but they require hardware modifications, these are (a) a serial port, and (b) JTAG cable. These are outside the scope of this document, please visit the openWrt wiki site for more information (http://wiki.openwrt.org)

Bootwait

Bootwait is a parameter that can be set in the on the router, which will cause the firmware to wait for approximately 4 seconds waiting for a tftp file transfer, before loading the openWrt image.from flash memory. This means that if you set “boot_wait=on” then in almost all cases you can use tftp to copy a new image to the router and restart the installation.

In my view, boot_wait should always be enabled, the extra 4 second wait during boot is worth it.

If you attempt to tftp an image to the router during “boot_wait”. The tftp server is running from the firmware. This is NOT the tftp process loaded from the openWrt flash image. Remember that openWrt has not loaded yet. Because we are accessing the firmware tftp server and the firmware doesn’t know the IP address that you have configured for your router, it will use a default address of 192.168.1.1/24. Make sure that you use 192.168.1.1 as the tftp destination and that your workstations address is in the same subnet, 192.168.1.2 is a good address for your workstation. And remember that you only have 4 seconds to start the tftp transfer. You may have to try a couple of times.

If you find that tftp asks for a password then you have missed the 4 second window to access the firmware tftp server, openWrt has booted, and you have accessed the openWrt tftp server. When this happens, reboot the router and try again.

Failsafe

Failsafe mode, is entered using the reset button on the back of router.
To enter failsafe mode you have to wait until openWrt loads. This means that you will have to wait until the DMZ light on the front of the router is lit. As soon as the DMZ indicator lights up, press the reset button for about 2-3 seconds. If done correctly the DMZ light will start flashing 3 times (rapidly) each second.

⚠ NOTE: Holding the reset button *before* the DMZ LED turns on (i.e. when the bootloader is still running) can reset the NVRAM parameters on some models.

Once you have entered failsafe mode the router will have the following characteristics

- The WAN is not brought up. – no internet access
- The firewall is not loaded

- The LAN and Wifi are both brought up in a bridged configuration
- The bridge (LAN & WiFi) Ip address is set to 192.168.1.1
- The file system is read only
- Your VLAN configuration is NOT preserved, All LAN ports are placed in the vlan0
- The telnet daemon will be active, ssh is deactivated

Because all LAN ports are placed in the same VLAN, it doesn't matter which LAN port you connect to. (This was not the case with RC4 and ealier) or via wireless as the wireless interface is bridged with the LAN ports. Remember that the routers address is now 192.168.1.1/24.

Once in failsafe mode, if you are connecting via LAN(vlan0) or via wireless, DHCP may issue you an IP address, depending ion your previous configuration. If an address is not allocated, try setting one manually, remembering it needs to be in the same subnet. 192.168.1.2 Would be a good choice. You should then be able to telnet to the router. You will not be able to use ssh.

Once connected to the router, you will be able to change any inappropriate NVRAM parameters that are causing a problem. However you can't change any of the scripts as the file system is read-only. To modify the file system it need to be re-mounted read-write. To do re-mount the / file system, issue the following commands:

```
# re-mount the root file system read-write
cd /
/sbin/mount_root
```

Configuration

This document is based on the configuration shown in the following diagram. If you are not using the same configuration, you will need to make appropriate changes to the information provided.

There a 4 network segments

1. lan one port only connected to a separate switch
2. dmz three ports
3. wl0 the wireless network, connecting home PC's
4. adsl the link to the adsl modem
5. wan the internet, pppoe over the adsl network segment

IP addressing scheme used

openWrt Interface Name	**openWrt Interface device**	**Ports**	**CPU Interface**	**IPaddress**	**Subnet**
lan	vlan0	1	eth0	192.168.1.254	255.255.255.0
wan	ppp0		eth0	dynamic	dynamic
dmz	vlan2	2 3 4	eth0	192.168.12.1	255.255.255.0
wl0	eth1		eth1	192.168.0.1	255.255.255.0
adsl	vlan1	0 (internet)	eth0	10.0.0.1	255.255.255.0
loop	lo		lo	127.0.0.1	255.0.0.0

Through each of the configuration steps, you will need to reboot the router for them to take effect, you don’t need to reboot after each step. It is Ok to perform the entire configuration, and then reboot once you have completed.

NOTE: I have deliberately set the IP address of my lan(vlan0) address to 192.168.1.254, so to allow other routers (all wrt54gs) that I work on from time to time to use the 192.168.1.1 (default) address.

Set the VLANs

For this device I have selected to have 4 separate interfaces. For the firewall

(1) Internet WAN network – including access to the ADSL modem
(2) Home WiFi network – considered to be (very) insecure
(3) DMZ – to host services to be exposed to the Internet
(4) Secure LAN – to host business & development servers

In order to achieve this I will use the Internet port (port0) for the wan connection
I will not be bridging the WiFi network to LAN (vlan1) as I want these to be separate networks. This means that the device “br0” will not be used / configured, and the Wifi interface will be routed. The 4 Switch ports will be configured in two VALNs, for the secure LAN and the DMZ. This is achieved with the following NVRAM parameters

Note. Port 5 is included in all the VLANs as this is the trunk port connected internally to the CPU. The “*” in *vlan0ports* indicates that this is the VLAN that traffic without a VLAN tag will be considered to belong to.

The VLAN configuration is defined by the NVRAM parameters, which are used during boot up to configure the VLANs and plump the devices.

Add the following NVRAM parameters, editing as appropriate

```
#Make sure that the parameter that configures “br0” is not set
nvram unset lan_ifnames

# The Secure “LAN” use port 1
nvram set vlan0ports="1 5*"
nvram set vlan0hwname=et0

# The “Wan” using port 0
nvram set vlan1ports="0 5"
nvram set vlan1hwname=et0

# The “DMZ” using ports 2-4
nvram set vlan2ports=" 2 3 4 5"
nvram set vlan2hwname=et0

#Commit the changes
nvram commit
```

Configuring the local interfaces

Once the VLANs have been created we will have the following local interfaces, vlan0 (the WAN) will be configured separately in the following section:

The three local interfaces are:

(1) vlan1 – Secure LAN
(2) vlan2 – DMZ
(3) eth1 – Wifi/home

To configure these add the following NVRAM parameters, editing as appropriate Each interface is configured with a static IP address: The NVRAM parameters are used by various scripts at boot-up to configure the interfaces appropriately

lan is the name given to vlan0 (The secure net)

```
# configures the LAN interface:  device=vlan0

nvram set lan_ifname=vlan0
nvram set lan_proto=static
nvram set lan_ipaddr=192.168.1.254
nvram set lan_netmask=255.255.255.0

nvram commit
```

dmz is the name used for vlan2 (the DMZ)

```
# configures the DMZ interface: device=vlan2

nvram set dmz_ifname=vlan2
nvram set dmz_proto=static
nvram set dmz_ipaddr=192.168.2.1
nvram set dmz_netmask=255.255.255.0

nvram commit
```

wifi is the name used for eth1 (The wireless 802.11b/g, used for the home network)

```
# configures the WiFi interface:  device=eth1

nvram set wifi_ifname=eth1
nvram set wifi_proto=static
nvram set wifi_ipaddr=192.168.0.1
nvram set wifi_netmask=255.255.255.0

nvram commit
```

Configuring the WAN interface

The WAN interface can be either: STATIC, DHCP or PPPOE depending on your requirements. I have provided configurations for each of them

Setting the Hostname

First you may want to set the Hostname for your router. The default hostname is OpenWrt. You can get the hostname via the wan_hostname nvram parameter

```
# set the routers hostname

nvram set wan_hostname=fw

nvram commit
```

WAN – STATIC

```
# configures the WAN with a static IP address:  device=vlan1

nvram set wan_ifname=vlan1
nvram set wan_proto=static
nvram set wan_ipaddr=10.0.0.1
nvram set wan_netmask=255.255.255.0

nvram commit
```

WAN – DHCP

```
# configures the WAN with a dynamic DHCP IP address:  device=vlan1

nvram set wan_ifname=vlan1
nvram set wan_proto=dhcp

nvram commit
```

WAN – PPPOE

When using PPPOE, another interface “ppp0” is created once the link is established. The ppp0 device is bound to the WAN interface “vlan1”

Also, if you have a configuration similar to mine, you will want to be able to access the admin interface of your ADSL modem from behind the Linksys router, ie: avoid plugging and unplugging cables etc. To do this I have statically assigned a IP address to the vlan1 interface in the same address space as the ADSL modem. My ADSL modem uses a default of 10.0.0.138/8, so I have chosen to use 10.0.0.1 as the IP address for the vlan1 interface.

Setup ADSL net on vlan1

Note: This section is optional, only required if you want to access the adsl modem over the router

```
#Configure vlan1 to allow access the ADSL modem

nvram set adsl_ifname=vlan1
nvram set adsl_proto=static
nvram set adsl_ipaddr=10.0.0.1
nvram set adsl_netmask=255.0.0.0

nvram commit
```

Set up the PPPOE interface

```
# Configure the PPP interface, bound to vlan1
nvram set wan_ifname=ppp0
nvram set wan_proto=pppoe

nvram set pppoe_ifname=vlan1

nvram set ppp_idletime=5
nvram set ppp_mtu=1492
nvram set ppp_username=”Your User Name”
nvram set ppp_passwd=”Your Password”

nvram commit
```

Setting the 802.11 parameters

The physical WiFi interfaces (eth1) was setup earlier (see configuring the local interfaces) This section deals with the configuration of the 802.11 link layer

I had a bit of trouble getting things working. At some point realised the following parameters need to be cleared. I have included the commands to clear them here just to make sure that they don’t cause trouble latter. They may not be set in a default installation, but it doesn’t hurt to be sure that they are not set

Clear the following NVRAM parameters:

```
# Clear these parameters

nvram unset wl0_country
nvram unset wl_country
nvram unset wl_country_code
nvram unset wl_chan_list
nvram unset wl_channel

nvram commit
```

Set the basic WiFi parameters
This is sufficient to get the WiFi working with out any security

```
# Basic WiFi parameters

nvram set wl0_ifname=eth1
nvram set wl0_ssid=”Insert your SSID”
nvram set wl0_channel=”Insert you channel number here”
nvram set wl0_mode=ap
nvram set wl0_infra=1
nvram set wl0_closed=0
nvram set wl0_country_code=All

nvram commit
```

Antenna Diversity

```
# set antenna diversity
nvram set wl0_antdiv=3
nvram commit
```

Using WPA

If you are using WPA, you cannot use WEP. You should select one or the other. Alternatively you could run no encryption or authentication, but that would not be advisable.

Set the following parameters if you are planning to use **WPA**.
Edit as necessary to meet your requirements. The parameters that you may need to update are marked in bold.

```
# WPA WiFi parameters

nvram set wl0_ifname=eth1
nvram set wl0_ssid=”Insert your SSID”
nvram set wl0_channel=”Insert you channel number here”
nvram set wl0_mode=ap
nvram set wl0_infra=1
nvram set wl0_closed=0
nvram set wl0_country_code=All
nvram set wl0_wpa_psk=”Your WPA password goes here”
nvram set wl0_wpa_gtk_rekey=3600
nvram set wl0_auth_mode=psk
nvram set wl0_wep=disabled
nvram set wl0_crypto=aes

nvram commit
```

NOTE: if using WPA, you will also need to install the “nas” package. Instructions of doing so are provided in the second stage “advanced configuration” (I’m not sure that this is true for White Russian 0.9)

Using WEP

If you are using WEP, you cannot use WAP. You should select one or the other. Alternatively you could run no encryption/authentication, but that would not be advisable.

NOTE: in my configuration I use WEP as I have several older 802.11b cards that don’t have drivers that support WAP

Use these parameters to if using **WEP**, changing as necessary
Parameters that you will probably need to change are marked bold

```
# WEP WiFi parameters

nvram set wl0_ifname=eth1
nvram set wl0_ssid=”Insert your SSID”
nvram set wl0_channel=”Insert you channel number here”
nvram set wl0_mode=ap
nvram set wl0_infra=1
nvram set wl0_closed=0
nvram set wl0_country_code=All
nvram set wl0_crypto=aes
nvram set wl0_auth_mode=wep
nvram set wl0_wep=enabled
nvram set wl0_wep_bit=64
nvram set wl0_key1=”Your WPA key goes here”

nvram commit
```

Applying MAC address restrictions

Because I ended up using WEP, I also wanted the extra protection assured by restricting the MAC address. This is achieved by setting the following parameters

```
# WiFi MAC address restriction

nvram set wl0_macmode=allow
nvram set wl0_maclist=”space separated MAC addrs xx:xx:xx:xx:xx:xx”

nvram commit
```

Note: if you have several machines, add each mac address with a space separating them. Do not add commas or any other form of separation characters. Also the mac address MUST be in hex format “xx:xx:xx:xx:xx:xx”

Basic Firewall configuration

The basic firewall configuration is contained in the start-up script /etc/init.d/S45firewall. This script requires some minor modification to cater for the extra VLAN's that I have created.

The modifications do the following

(1) extract the devices associated with each VLAN
(2) LAN (ports 3 & 4) are permitted anywhere, inc ADSL
(3) WIFI is only permitted to access the Internet
(4) DMZ is only permitted to access the Internet
(5) All traffic is NATed behind the WAN IPaddress

The following is the modified start-up script **/etc/firewall.usr**

```
#!/bin/sh
. /etc/functions.sh

WAN=$(nvram get wan_ifname)
LAN=$(nvram get lan_ifname)
DMZ=$(nvram get dmz_ifname)
ADSL=$(nvram get adsl_ifname)
WIFI=$(nvram get wifi_ifname)


iptables -F input_rule
iptables -F output_rule
iptables -F forwarding_rule
iptables -t nat -F prerouting_rule
iptables -t nat -F postrouting_rule

### BIG FAT DISCLAIMER
## The "-i $WAN" is used to match packets that come in via the $WAN
interface.
## it WILL NOT MATCH packets sent from the $WAN ip address -- you
won't be able
## to see the effects from within the LAN.

### Open port to WAN
## -- This allows port 22 to be answered by (dropbear on) the router
# iptables -t nat -A prerouting_rule -i $WAN -p tcp --dport 22 -j
ACCEPT
# iptables        -A input_rule      -i $WAN -p tcp --dport 22 -j
ACCEPT

### Port forwarding
## -- This forwards port 8080 on the WAN to port 80 on 192.168.1.2
# iptables -t nat -A prerouting_rule -i $WAN -p tcp --dport 8080 -j
DNAT --to 192.168.1.2:80
# iptables        -A forwarding_rule -i $WAN -p tcp --dport 80 -d
192.168.1.2 -j ACCEPT

### DMZ
## -- Connections to ports not handled above will be forwarded to
192.168.1.2
# iptables -t nat -A prerouting_rule -i $WAN -j DNAT --to 192.168.1.2
# iptables        -A forwarding_rule -i $WAN -d 192.168.1.2 -j ACCEPT


# NAT outgoing traffic on the ADSL interface to the localnet address.
# The ADSL modem doesn't have a default gateway, so cant reach other
# nets. This will allow the ADSL modem to be accessible from the LAN
```

```
iptables -t nat -A postrouting_rule -o $ADSL -j MASQUERADE
iptables        -A forwarding_rule -i $LAN -o $ADSL -j ACCEPT

# Permit Other Nets access to Internet

iptables        -A forwarding_rule -i $DMZ -o $WAN -j ACCEPT
iptables        -A forwarding_rule -i $WIFI -o $WAN -j ACCEPT

# Permit LAN to access other nets

iptables        -A forwarding_rule -i $LAN -o $DMZ -j ACCEPT
iptables        -A forwarding_rule -i $LAN -o $WIFI -j ACCEPT
```

Starting the interfaces

The NVRAM “parameter “ifup_interfaces” is now used determine the specify the interfaces that should be started when the router boots. I want to start the following interfaces “lan”, “wan”, “wifi”, “adsl” and “dmz”

```
# Specify the interfaces to start

nvram set ifup_interfaces=”lan wan wifi adsl dmz”

nvram commit
```

DNS and DHCP servers

DNS and DHCP services are provided by the “dnsmasq” server. This server reads the configuration file “/etc/dnsmasq.conf” by default. The configuration file that I am using also (optionally) reads the file /etc/hosts, and /etc/ethers.

Sample /etc/hosts

Add host entries to “/etc/hosts” as required, feel free to use aliases etc. The DNS server will read the contents of the hosts file as serve them out locally.

The following is a **sample** of my **/etc/hosts** file

```
##################################################################
# NOTE:  dnsmasq automatically adds the domain martin.cc
#        for all hostnames & aliases without a domain
#################################################################
```

```
############################################
# IP address used on the Linksys
############################################
127.0.0.1       localhost fw_loopback
192.168.0.1     fw_wifi fw
192.168.1.254   fw_lan  fw sip
192.168.2.1     fw_dmz  fw
10.0.0.1        fw_adsl fw

############################################
# IP address of the admin interface for the ADSL modem
############################################
10.0.0.137      modem   adslmodem adsl_modem dlink

############################################
# Static devices on the DMZ
############################################
192.168.2.140   eagle   sbs  svn  www ftp wpad www.voicetime.com.au
products docs doc gallery music trac
192.168.2.149   printer printer-dmz
192.168.2.130   snowbird

############################################
# Static devices on the LAN
############################################
192.168.1.2     beast_wireless  beast

192.168.1.21    beast_ethernet
192.168.1.22    duck            slimserver
192.168.1.23    robin           s2sy-tts-0
192.168.1.24    parrot
192.168.1.25    budgie

192.168.1.99    ata
192.168.1.100   phone
192.168.1.101   emu



############################################
# Static devices on the WIFI
############################################
192.168.0.101   taylor  chicks
192.168.0.102   jacquie chicken
192.168.0.103   breana

############################################
# Other usefull aliases
############################################

############################################
# Davids servers - other side of VPN/GRE
############################################
192.168.102.2   obelix  obelix.home.net

# This is dave's external IP
203.206.128.239 varnes.net
```

Sample /etc/ethers

***NOTE**: I am no longer using /etc/ethers. I am now putting MAC-to-IP config in the /etc/dnsmasq.conf file. I get a bit more flexibility by doing this, and I don't need another config file.*

To provide static IP address based on MAC or Hostname do the following
Add MAC address and IP's to "/etc/ethers" (make sure that it is readable: ie chmod +r /etc/ethers)

NOTE: The documentation indicates that you can add the hostname to the ethers file. However I found that if a host name is included, that the line from /etc/ethers will not be used. (This may be corrected in White Russian 0.9. I haven't tested it)

The following is a sample from my old **/etc/ethers** file

```
# Mac Address           IP address
# Wireless
00:0C:F1:47:3E:93       192.168.1.2
# Ethernet
00:0D:60:F8:D0:19       192.168.0.25
```

Sample /etc/dnsmasq.conf

In my configuration I wanted DHCP allocated IP address on the **wifi**, **lan**, **dmz and adsl** networks. To achieve this I have used the following for my "/etc/dnsmasq.conf" file

Sample **/etc/dnsmasq.conf**

```
# filter what we send upstream
domain-needed           # forward DNS lookups only if they have a domain
bogus-priv              # fake private IP reverse lookups
filterwin2k             # don't forward win2k LDAP requests
localise-queries        # order DNS responses by local interface


# Set the domain name and allow expansion of hostnames
domain=martin.cc        # domain sent vi DHCP and used to expand hostnames
expand-hosts            # add the domainname to simple names in hosts

# Listen only on these interfaces (dns and dhcp service)
interface=vlan0         # LAN
interface=vlan1         # ADSL
interface=vlan2         # DMZ
interface=eth1          # WIFI

# Provide DNS but no DHCP on these interfaces
no-dhcp-interface=tun0

# Dont listen on these interfaces
#except-interface=ppp0
#except-interface=tun0

# Listen for other IP address that may be allocated
dhcp-authoritative

# Specify special DNS servers
# server=/domain/ns-server-ip#port@local-source-ip#port
#local=/martin.cc/     # Sortcut format for local domains, no forwarding
```

```
server=/home.net/192.168.5.2
server=/varnes.net/192.168.5.2

# Disable negative caching - unresolved names are tried again
no-negcache

# Set the location of files
resolv-file=/tmp/resolv.conf.auto
dhcp-leasefile=/tmp/dhcp.leases

# enable dhcp (start,end,netmask,leasetime)
#dhcp-range=[[net:]network-id,]<start-addr>,<end-addr>[[,<netmask>],<broadcast
>][,<default lease time>]
## Wifi
dhcp-range=eth1,192.168.0.150,192.168.0.200,255.255.255.0,12h
## LAN
dhcp-range=vlan0,192.168.1.150,192.168.1.200,255.255.255.0,12h
## ADSL Modem (Quarantine)
#dhcp-range=vlan1,10.0.0.150,10.0.0.200,255.255.255.0,12h
## DMZ
dhcp-range=vlan2,192.168.2.150,192.168.2.200,255.255.255.0,12h

# use /etc/ethers for static hosts; same format as --dhcp-host
# <hwaddr> <ipaddr>
###Not using ethers
#read-ethers

# Static IP address
# dhcp-host=[[<hwaddr>]|[id:[<client_id>][*]]][,net:<netid>][,<ipaddr>]
[,<hostname>][,<lease_time>][,ignore]
dhcp-host=00:14:a4:51:bf:b8,beast,192.168.1.2
dhcp-host=00:13:02:ac:64:7e,chicken


# other useful options:
# default route(s): dhcp-option=vlan0,3,192.168.1.1,192.168.1.2
#    dns server(s): dhcp-option=vlan0,6,192.168.1.1,192.168.1.2
```

Starting the modified “dnsmasq” configuration

Since we have a very specific “dnsmasq configuration” file, the standard startup script, “/etc/init.d/S50dnsmasq” is no longer appropriate. The default script assumes that dhcp will only be server over the “lan” network, however I wanted to use dhcp over all my internal interfaces. The default startup script extracts parameters from nvram, if none are available it assumes some sensible defaults and then builds a augments to pass on the command line. I don’t want these, so I have replaced the file “/etc/init.d/S50dnsmasq”, with my own

Cut and past the following to the command line, a then edit /etc/init.d/S50dnsmasq” if required

```
# No need to save the original S50dnsmasq start-up script
# it is always available in squashfs /rom/etc/init.d S50dnsmasq

# create a new script
cat <<!EOF > /etc/init.d/S50dnsmasq
#!/bin/sh

case $1 in
```

```
start)
        dnsmasq && {
        # use dnsmasq for local dns requests
        rm -f /tmp/resolv.conf
        echo "nameserver 127.0.0.1" > /tmp/resolv.conf
        echo "search martin.cc" >> /tmp/resolv.conf
        }
        ;;

esac

!EOF

# make sure that it is executable
chmod 744 /etc/init.d/S50dnsmasq
```

Configuring Remote Syslog

“syslogd” and “klogd” are both started from /etc/init.d/rcS, which in turn is started by “init”. With RC4+, the “/etc/init.d/rcS” script now checks for an nvram variable called “log_ipaddr”. If this is set to a valid IP address, syslog will be stared doing both local and remote logging.

If you are going to do remote logging to another syslogd server, you will also need to configure the remote syslogd server to accept incoming messages. This is usually done by adding a “-r” argument to the command line in the /etc/init.d start-up script

To enable remote do the following

```
nvram set log_ipaddr=192.168.2.140
nvram commit
```

NOTE: currently this will only accept and IP address. Syslogd will accept a hostname as an augment, and it will resolve it from /etc/hosts. However the /etc/init.d/rcS check for a valid IP address not a hostname. This would need to be changed.

Advanced Configuration

NOTE 1: *to install the packages, you will need to ensure that you are able to access the internet from your router. If you cannot reach the internet, go back and check/troubleshoot your configuration*

NOTE 2: *make sure that you have rebooted your router, before you start this section, to ensure that any configuration changes you may have made earlier have taken effect*

Update package sources

I want to use some to the kamikaze apps that have been backported to White Russian 0.9. To do this add a new src line to the **/etc/ipkg.conf** file

```
# Add 0.9 backports to ipkg.conf
echo “src backports http://downloads.openwrt.org/backports/0.9” >>
/etc/ipkg.conf
```

Update the base packages

Packages are installed using the “ipkg” utility.
Run the following commands to get update the package list and then perform an upgrade of any of the in stalled packages

```
# Get the latest package lists
ipkg update

# upgrade any installed packages
ipkg upgrade
```

Install “nas”

In order to get WPA to operate you will also need to install the “nas” package:. The following command will install the “nas” package

I think, in White Russian 0.9, you have several options. The “nas” package is from Broadcom. I think that OpenWrt have written there own called “wpa-supplicant”. You can use that instead.

You only need this if you are going to use WPA

```
# Network Authentication Service
# Required if you are using WPA
ipkg install nas
```

Install “tcpdump”

“tcpdump” is an extremely useful package. Issue the following to get it installed

```
# This pkg Allows packet snooping & problem diagnosis
ipkg install tcpdump
```

NOTE: This will also install “libcap” at the same time, it is a pre-requisite.

Install and configure “openntpd”

“openntpd” is an NTP server implementation that will keep the local time of the router synchronised with external time sources. This is particularly important if the log files of the router such as syslog are to be used to diagnose issues.

The following command will install the opoenntpd package

```
# NTP time server
ipkg install openntpd
```

For my location, I needed to set TZ (the time-zone environment variable) to:
TZ=EST-10EDT-11,M10.1.0/02:00:00,M3.4.0/03:00:00
Your location will require a different TZ specification.
The format for TZ is: std-[offset[dst-[offset][,start[/time],end[/time]]]]

The TZ variable needs to be set in two locations

1. /etc/profile
2. /etc/TZ

The following commands will create the file **“/etc/TZ**”. This file is used by other utilities to get the correct time zone, and add the time zone to the bottom of /etc/profile

Cut and past to the command line, then edit /etc/TZ and amend the timezone information for your area. This is the correct TZ for Australia/NSW

```
# Create the file /etc/TZ
echo “EST-10EDT-11,M10.4.0/02:00:00,M3.4.0/03:00:00” > /etc/TZ

# Set the correct permissions
chmod 644 /etc/TZ
```

```
# Append to /etc/profile
cat <<!EOF >>/etc/profile

# The timezone variable
# Start of daylight savings is last Sunday in October @ 2am
# End of daylight savings is last Sunday in March @ 3am
if [ -f /etc/TZ ]
then
    export TZ=”\$(cat /etc/TZ)”
fi
!EOF
```

The CPU time is maintained by the NTP demon once it starts. While the default NTP configuration files are adequate to run the process, there is however an issue in that the NTP daemon may not be able to reach an external time source until the WAN interface has started (In my case this can take several minutes, sometimes as much as 10 minutes). The NTP daemon appears to only be able to step the time from 0 to the real date & time on its first update. Since it may not be able to reach a time source immediately it will not step the date & time correctly

To overcome this problem, I use “cron” to run a small sequence of commands to detect if date has been set, and if not, to use rdate to synchronise the date & time. See the following section on configuring cron.

In “cron” I have added the following:

```
      # The date is not set if y=2000, so try and set it
      0-59/2 * * * * ([ "$(date +%y)" = "00" ] && rdate
rdatesvr.domain.com)
```

This will check the date every 2 mintues and if the year is 2000 will use rdate to get the current time from a specified rdate server.

For more information of setting up cron, see the following section “Configuring Cron”

Configuring DynDNS IP address updates

NOTE: White Russian 0.9 supplies a package called “ez-ipupdate” which can be used in place of the “dyndnsupd” script used in this documentation. I have not used it only because I am not yet familiar with it

In order to use DynDNS you will need to install the “dyndnsupd” script (provided in utilities section), as well as the mini-sendmail package .(documentation provided in the Appendix)

To install the mini-sendmail package issue the following command:

```
# Install mini-sendmail
ipkg install mini-sendmail
```

The dyndns script is called from several locations

(1) Hotplug to update dyndns with the wan interface address when the interface comes up. (see hotplug setup later in this section)

(2) From cron to monitor the current ip address as well as the force an update every week to ensure that the account remains active. (see setting up cron in the following section)

Parameters at the beginning of the scrip will need to be customised to your configuration

Configuring CRON

Configuration files for cron are usually located in the directory /var/spool/cron/crontabs, however the /var directory is sym-linked to the /tmp (ramdisk) and its contents are destroyed at each reboot.

To overcome this openWrt has placed the crontabs directory under the /etc directory (.i.e: /etc/crontabs)

The following script will create my default root crontab file in the root directory that will perform the following:

1. Reboot the router everyday to avoid
2. Check if the year is “00” and if so, use “rdate” to sync the date& time
3. Check our IPaddress to ensure that it is set correctly with DynDNS

NOTE: You will note that in the script that I an no longer rebooting the router or using dynddnsupd. Things are now very stable and I now have a fixed IP address.

Cut and paste the following to the command line, then edit “/etc/crontabs/root” as needed

```
# create the root crontab file
cat <<!EOF >/etc/crontabs/root
# reboot at every night at 5am in the morning
# 0 5 * * * (ifdown wan; /sbin/reboot)

# See if we need to do an DynDNS update .  check every 15 minutes
# 0,15,30,45 * * * * /etc/bin/dyndnsupd

# The date is not set if y=2000, so try and set it
0-59/2 * * * * ([ “\$(date +%y)” = "00" ] && rdate
yourRdateSvr.domain.com)
!EOF

# Set appropriate permissions
chmod 600 /etc/crontabs/root
```

(openswan) – IPSEC

This was taken from my RC5 installation. I am no longer using IPSEC/OpenSwan, as I found an MTU issue with the ipsec interface. I am now using GRE tunnels, with SSH for sensitive stuff

In order to use ipsec install the following packages

```
# install openswan
ipkg install openswan
```

Note: this will also install the dependences:
kmod-openswan
libgmp
ip

if you have backup copies, then restore the files /etc/ipsec.secrets and /etc/ipsec.conf making sure that /etc/ipsec.secrets has appropriate permissions
i.e. chmod 400 /etc/ipsec.secrets

if you do NOT have backup copies, you will need to run the command

```
ipsec newhostkey --output /etc/ipsec.secrets
```

This will create new keys that you will need to distribute to you IPSec partners
The file contains both your public and private keys. Make sure that the permissions are correct and that you only ever give your public key.

The following is the contents of my **/etc/ipsec.conf** file:

```
# /etc/ipsec.conf - Openswan IPsec configuration file
# RCSID $Id: ipsec.conf.in,v 1.15.2.2 2005/11/14 20:10:27 paul Exp $

# This file:  /usr/share/doc/openswan/ipsec.conf-sample
#
# Manual:     ipsec.conf.5

version 2.0     # conforms to second version of ipsec.conf
specification

# basic configuration
config setup
        myid=@my.domain.com
        # plutodebug / klipsdebug = "all", "none" or a combation from
below:
        # "raw crypt parsing emitting control klips pfkey natt x509
private"
        # eg:
        # plutodebug="control parsing"
        #
        # Only enable klipsdebug=all if you are a developer
        # NAT-TRAVERSAL support, see README.NAT-Traversal
        # nat_traversal=yes
        # virtual_private=
%v4:10.0.0.0/8,%v4:192.168.0.0/16,%4:172.16.0.0/12
        # interfaces=%defaultroute

# Add connections here

# sample VPN connection
#sample#        conn sample
#sample#                # Left security gateway, subnet behind it,
next hop toward right.
#sample#                left=10.0.0.1
#sample#                leftsubnet=172.16.0.0/24
#sample#                leftnexthop=10.22.33.44
#sample#                # Right security gateway, subnet behind it,
next hop toward left.
#sample#                right=10.12.12.1
#sample#                rightsubnet=192.168.0.0/24
#sample#                rightnexthop=10.101.102.103
#sample#                # To authorize this connection, but not
actually start it, at startup,
#sample#                # uncomment this.
#sample#                #auto=start

# VPN connection to David
conn voicetime
        # use RSA key authentication
        authby=rsasig
        # use compression
        compress=yes
        # Left security gateway, subnet behind it, next hop toward
right.
        left=my.domain.com
        leftsubnet=192.168.1.0/21
        leftrsasigkey=<my public key from /etc/ipsec.secrets>
        leftid=@my.domain.com
        #leftupdown=/usr/lib/ipsec/_updown
        leftnexthop=%defaultroute
        # Right security gateway, subnet behind it, next hop toward
left.
        right=david.domain.org
        rightsubnet=192.168.100.0/21
        #rightupdown=/usr/lib/ipsec/_updown
        rightnexthop=%defaultroute
        rightrsasigkey=<davids public key from /etc/ipsec.secrets>
        rightid=@david.domain.org
        # To authorize this connection, and start it, at startup,
        auto=start

#Disable Opportunistic Encryption
include /etc/ipsec.d/examples/no_oe.conf
```

NOTE 1: a blank line in the file indicates the end of a section, you cannot add blank lines to aid with formatting and readability.

NOTE2: I am starting openswan via hotplug when the ppp0 interface starts. So I have renamed the /etc/init.d/S60ipsec start-up script to /etc/init.d/x60ipsec

```
# move the S60ipsec start-up script
[ -f /etc/init.d/S60ipsec ] && mv /etc/init.d/S60ipsec
/etc/init.d/x60ipsec
```

GRE tunnel

In order to use a GRE tunnel you will need to install the “kmod-gre” and “ip” packages

```
# install packages for GRE tunnels
ipkg install kmod-gre
ipkg install ip
```

I have used the following script the in **/etc/bin/gre** to start and stop the gre tunnel. It can only be run after the WAN interface is up, so I don’t launch it as a rc start-up script. I launch it from a hotplug script when the WAN interface cones up/down

```
#!/bin/sh

REMOTE_ENDPOINT="203.206.128.239"        # The IP of the far end of the
tunnel
LOCAL_ENDPOINT="203.206.172.245"         # The IP of the local end of
the tunnel
TUNNEL_DEV="tun0"                        # The name used to identify
the tunnel interface
TUNNEL_IP="192.168.5.1/30"               # The IP address to use on the
tunnel interface
BIND_DEV="ppp0"                          # The interface that the
tunnel will bind to on the local end

# The subnets that we should route over the GRE Tunnel
ROUTES="192.168.100.0/24 192.168.101.0/24 192.168.102.0/24
192.168.104.0/24"


case $1 in

start)
        # load the GRE kernel module
        insmod ip_gre

        # create the GRE interface
        ip tunnel add ${TUNNEL_DEV} mode gre remote ${REMOTE_ENDPOINT}
local ${LOCAL_ENDPOINT} dev ${BIND_DEV} ttl 255

        # bring the link up
        ip link set ${TUNNEL_DEV} up

        # give it an address
        ip addr add ${TUNNEL_IP} dev ${TUNNEL_DEV}

        # add any required routes
        [ -z "${ROUTES}" ] ||  for ROUTE in ${ROUTES}
        do
                ip route add ${ROUTE} dev ${TUNNEL_DEV}
        done
        ;;

stop)
        ip link set ${TUNNEL_DEV} down
        ip tunnel del ${TUNNEL_DEV} mode gre remote ${REMOTE_ENDPOINT}
local ${LOCAL_ENDPOINT} dev ${BIND_DEV}
        # rmmod ip_gre

        ;;

*)
        echo "Usage: gre start|stop"
        ;;

esac
```

(pptpd) – Windows VPN

In order to use pptpd install the following packages

```
# install pptpd
ipkg install pptpd
```

Note: this will also install the dependences:
kmod-crypto
kmod-mppe

if you have backup copies, there restore the files /etc/ppp/options.pptpd and /etc/ppp/chap-secrets

Make sure that /etc/ppp/chap-secrets has appropriate permissions
i.e. chmod 400 /etc/ppp/chap-secrets

The contents of /etc/ppp/options.pptpd should look something like

```
#debug
#logfile /tmp/pptp-server.log
# The following the IP address that clients will see
192.168.4.1:
auth
name "pptpdSvrName"
lcp-echo-failure 3
lcp-echo-interval 60
default-asyncmap
mtu 1482
```

```
mru 1482
nobsdcomp
nodeflate
#noproxyarp
#nomppc
mppe required,no40,no56,stateless
require-mschap-v2
refuse-chap
refuse-mschap
refuse-eap
refuse-pap
#ms-dns 172.16.1.1
#plugin radius.so
#radius-config-file /etc/radius.conf
```

Make sure that you set the IP address that you want the clients to see as the next hop router, and specify a name for your pptpd sever

The contents of /etc/ppp/chap-secrets should look something like

```
#USERNAME       PROVIDER        PASSWORD        IPADDRESS
cmartin         pptdpSrvName    pass            192.168.4.20
david           pptdpSrvName    pass            192.168.4.21
```

Add entries for each user as required. By specifying a unique ID for each user, you will be able to customise the firewall rules based in the users allocated IP address.

Using Hotplug scripts

Where are several events you may want to take place when an interface is brought up or down. In the past this had to be configured in various scripts depending if the interface is using dhcp, pppoe or other configuration. Fortunately with RC4+, the guys at openWrt have implemented hotplug, which means that these events can be handled in the one location independent of the interface configuration. This makes things much easier.

All interface events are manages by scripts in the “/etc/hotplug.d/iface” directory.

When the interface is brought up (ACTION=ifup), I have a few things that I need done

(1) Reload the filrewall rules (see the section on using firewall builder)
(2) Reload the qos rules
(3) Update DynDns with my IP address
(4) Prime the router clock with the correct time

When the interface is brought down (ACTION=ifdown)

(1) Reload the filrewall rules (see the section on using firewall builder)

I have used the following script on my routers.
Cut and past to the command line, and then edit he file /etc/hotplug.d/iface/wan and amend as appropriate for your configuration.

```
mkdir /etc/hotplug.d/iface
cat <<!EOF > /etc/hotplug.d/iface/wan

if [ "\${INTERFACE}" = "wan" ]
then
        case \$ACTION in

        "ifup")
                # Make sure that time has been updated
                rdate rdateSvr.domain

                # update DynDNS
                # /etc/bin/dyndnsupd

                # start ipsec
                # ipsec setup start

                # start GRE
                # /etc/bin/gre start

                # Reload the firewall rules
                # /etc/init.d/S36fwbuilder start

                # start pptp
                # I stop/start pptp in sync with the WAN interface to
                # ensure that the WAN is always ppp0
                # /etc/init.d/x50pptpd start

                /usr/bin/logger -p INFO "wan up"
                ;;

        "ifdown")
                # stop pptp
                # /etc/init.d/x50pptpd stop

                # stop GRE
                # /etc/bin/gre stop

                # stop ipsec
                # ipsec setup stop

                # Reload the firewall rules
                # /etc/init.d/S36fwbuilder start

                /usr/bin/logger -p INFO "wan down"
                ;;

        *)
                # we should not get here.  log the event
                /usr/bin/logger -p ERROR "HOTPLUG: unhandled ACTION in
$0"
                ;;
        esac
fi
!EOF

chmod +x /etc/hotplug.d/iface/wan
```

Disable HTTP Server

I don’t use the HTTP interface. You never know when it may be useful, so for now I just disable it in /etc/init.d. You can’t uninstall it as it is on the squashfs Read Only file system. If you want to same the flash space you will have to build your own *.bin files without the http server. Or build JFFS binaries that will allow you to remove HTTP

```
# Disable the HTTP server
mv /etc/init.d/S50httpd /etc/init.d/x50httpd
```

Asterisk

Initially I used the version of asterisk from the OpenWrt packages.
However I recently hosed my router, and lost the config etc

So I am now using version I built myself with patches that avoid unnecessary transcoding, enabling me to run more calls with less CPU utilisation.
The standard SIP channel in Asterisk does a poor

Installing Asterisk

then issue the following commands

```
# Update package information
ipkg update

# install Asterisk
ipkg install asterisk
```

This will also install the packages libncurses and libpthread

Asterisk stores registry information in a small DB1 database. This is presently located in /usr/lib/asterisk/astdb, and need to to be moved/linked to the ramfs

```
# move astdb

# make sure a file exists
touch /usr/lib/asterisk/astdb
mkdir -p /var/spool/asterisk
mv /usr/lib/asterisk/astdb /var/spool/asterisk
ln -s /var/spool/asterisk/astdb /usr/lib/asterisk/astdb
```

Patch a typo in /etc/asterisk/modules.conf

The line:

```
      noload => app_setdrnis.so ; Set RDNIS Number
```

should be :

```
      noload => app_setrdnis.so ; Set RDNIS Number
```

To make asterisk start, issue the following

```
# link asterisk to startup rc
cd /etc/init.d
ln -s asterisk S90asterisk
```

Edit:

```
/etc/asterisk/sip.conf
/etc/asterisk/extensions.conf
```

NOTE: *******
Asterisk SIP isn’t supported by the dnsmgr, so IP address are still resolved at startup

Using Firewall Builder (fwbuilder)

Configuring openWrt

Restrictions

With the current release of openWrt you are not able to use “time” restrictions in a fwbuilder policy, as the iptables “time” module is not provided in the RC5 release. The openWrt developers have addressed this in the current kamakazie source, however unless you want to compile your own openWrt distribution, you will have to wait for the next major version.

Install packages

In order to use firewall builder on an openWrt router you will need to install the following packages

```
# install packages for fwbuilder
ipkg install ip
ipkg install iptables-mod-extra
ipkg install iptables-utils
```

Note: kmod-ipt-extra will also be installed, iptables-mod-extra is dependant on it

Load logging & limiting modules

Firewall builder needs the capability to perform logging from within iptables. To aceive this the module “ipt_LOG must be loaded. Also, if you plan to use rate limiting, you will also want to ipt_limit module. The best place to load these is in /etc/modules. Following is a copy of **/etc/modules** after editing

```
ipt_LOG
ipt_limit
wl
```

Create the fwbuilder script file

Firewall builder will copy a file from the workstation, to the router via ssh
I have used the **/etc/firewall.fw** as the location for the fwbuilder script. If the file doesn’t exit, it will be created, but with the wrong permissions.

Best to create an empty file and set the permissions on it manually

The following commands will create /etc/firewall.fw

```
# create script file for fwbuilder
touch /etc/firewall.fw
chmod 744 /etc/firewall.fw
```

Adding the pkill script

Under some conditions the script generated by fwbuilder will expect to be able to use a pkill command. This is not one of the commands available in openWrt. Create the "pkill" script as shown in the "Utilities" section. This will provide suitable functionality

The script generated by fwbulIder doesn't specify a path, so access to the script needs to be provided via the PATH variable specified in "/etc/profile"

Modify /etc/profile so that it resembles the following

```
#!/bin/sh
[ -f /etc/banner ] && cat /etc/banner

export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/etc/bin
export PS1='\u@\h:\w\$ '

alias less=more
alias vim=vi

arp() { cat /proc/net/arp; }
ldd() { LD_TRACE_LOADED_OBJECTS=1 $*; }
reboot() { ifdown wan 2>&1 >/dev/null ; /sbin/reboot; }

# The timezone variable
# Start if daylight savings is first Sunday in October @ 2am
# End of daylight savings is last Sunday in March @ 3am
if [ -f /etc/TZ ]
then
    export TZ="$(cat /etc/TZ)"
fi
```

Modify openWrt start-up script

Since we will be using the firewall script generated by fwbulilder, we don't need the default openWrt script **/etc/init.d/S35firewall**. However the there are several factors that need to be handled in any script that we do replace it with

(1) It must bypass the script if we are running in "FAILSAFE" mode
(2) In the event that there is a problem with the script that fwbuilder generates, it should default to loading to a safe firewall state
(3) From time to time you may need to stop the firewall that fwbuilder has started, and fall back to a safe firewall state. (i.e the ability to start / stop the firewall)
(4) The firewall script must also be run when ever the wan interface goes up of down. This done using the hotplug script shown ealier in this documentation, however this means that there may be multiple instances of the firewall script running at once. This will cause

problems with IP tables. For this reason a mechanism is needed that will ensure that only one copy is run at once.

(5) I don’t want to modify the S35firewall scrip, in order to make upgrading easier in the future

I have a second firewall script **/etc/init.d/S36fwbuilder** specifically for Fwbuilder with the following. It runs directly after the default S35firewall script. It seems a bit of a waste to fun S35firewall and then blow it away by running S36fwbuilder, but it does make upgrading/updating easier

```
#!/bin/sh

USER_RULES=/etc/firewall.fw
SPIN_LOCK=/etc/bin/spinlock


loadDefaultRules()
{
        echo "Loading default firewall rule set"

        . /etc/init.d/S35firewall start

}

#################################################################
#       START
#################################################################

case $1 in

"start")
        # Check if the required files are present
        if [ -f "$USER_RULES" ] && [ -x "$SPIN_LOCK" ]
        then
                # Attempt to load the rules
                if $SPIN_LOCK $USER_RULES
                then
                        echo "Custom firewall rules successfully
loaded"
                        exit 0
                else
                        echo "An error occured while loading the
custom firewall set"
                fi
        else
                echo "Unable to load custom firewall rule set"
                echo "Either \"$USER_RULES\" or \"$SPIN_LOCK\" are not
available or have incorrect permissions"
        fi

        # If we get here, then either the utilities are not present,
or the rules failed to load
        # So we will load the default set

        loadDefaultRules

        ;;
```

```
"stop")
        # Get rid of any custom rules by loading the default rules
        # Perhaps in the future this should block all traffic to the
internet.....
        # Or ... you could just unplug the router...
        # So I have opted to just load the default rules for now

        loadDefaultRules

        ;;

*)
        echo "Usage: $0 [start | stop ] "

        ;;

esac
```

make sure that it is exacutable

chmod +x /etc/init.d/S36fwbuilder

This script has a dependency on the script "**spinlock**" provided in the utilities section of this document.

Once you have created the script, edit the parameters at the top providing

(1) The location of the script created by fwbuilder.
(2) The location of the spinlock script.

Running the firewall script via hotplug

The reason that the fwbuilder script must be run whenver the wan interface changes state is that it contains lines that look like the following

```
      test -n "$i_ppp0" && \
            $IPTABLES -A Cid439539FE.0  -d $i_ppp0  -j Cid439539FE.1
```

Where "$ip_pp0" is the IP address of the ppp0 interface. However when the fwbuilder script is run, the ppp0 interface may not be up, and there for doesn't have an IP address, and as a result the rule is not loaded/linked.

The only way to ensure that the rules are loaded is to ensure that they are run whenever the interface is brought ip or down. It probably isn't too important to run the fwbuilder script when the interface goes down, but I 'm not sure of the security ramifications, so I run it when the interface goes down as well.

Since fwbuilder script is run when the router is started, and whenever the interface state changes, there is a real possibility that one more than one copy could be running at a time. For this reason I have used a utility called "spinlock" to ensure that only one copy is running at once.

See the earlier section on "Using Hotplug Scripts" and uncomment the lines that run the firewall rc/init script.

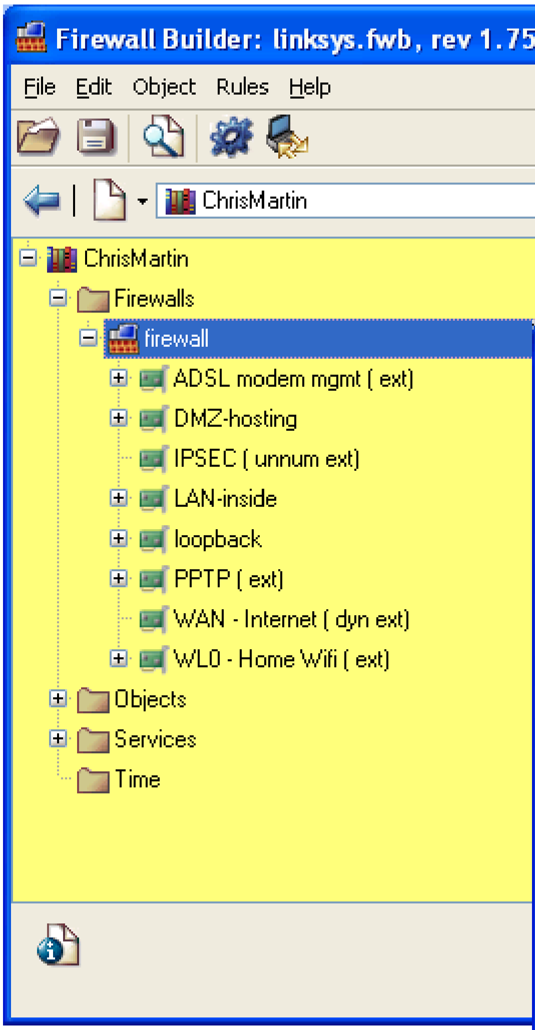
Configuring Firewall Builder

Download and install the appropriate version from firewall builder from:
http://www.fwbuilder.org

If you are using windows you will also need a copy of the putty ssh client (in particular plink and pscp)
http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

configure fwbuilder as per the following screens:

Once you have created a firewall object for the router and set up the interface that you require. Something similar to the diagram below



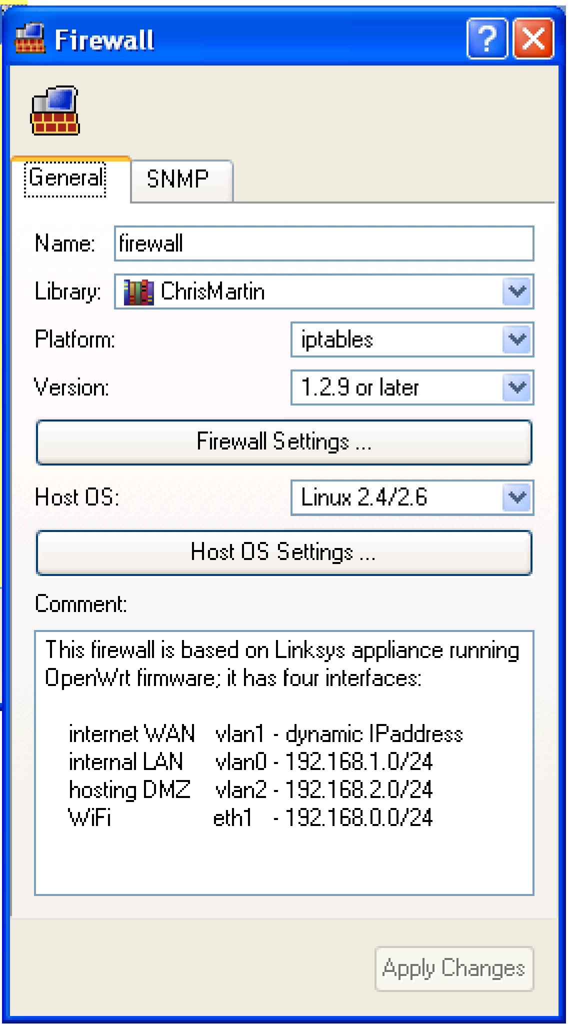
Firewall Builder: linksys.fwb, rev 1.75

ChrisMartin

- ChrisMartin
  - Firewalls
    - firewall
      - ADSL modem mgmt (ext)
      - DMZ-hosting
      - IPSEC (unnum ext)
      - LAN-inside
      - loopback
      - PPTP (ext)
      - WAN - Internet (dyn ext)
      - WL0 - Home Wifi (ext)
  - Objects
  - Services
  - Time

Make sure that you define all your interfaces.

Select the firewall object and chose edit from the menu. The following screen will appear



Firewall

Name: firewall

Library: ChrisMartin

Platform: iptables

Version: 1.2.9 or later

Host OS: Linux 2.4/2.6

Comment:

```
This firewall is based on Linksys appliance running
OpenWrt firmware; it has four interfaces:

  internet WAN   vlan1 - dynamic IPaddress
  internal LAN   vlan0 - 192.168.1.0/24
  hosting DMZ    vlan2 - 192.168.2.0/24
  WiFi           eth1  - 192.168.0.0/24
```

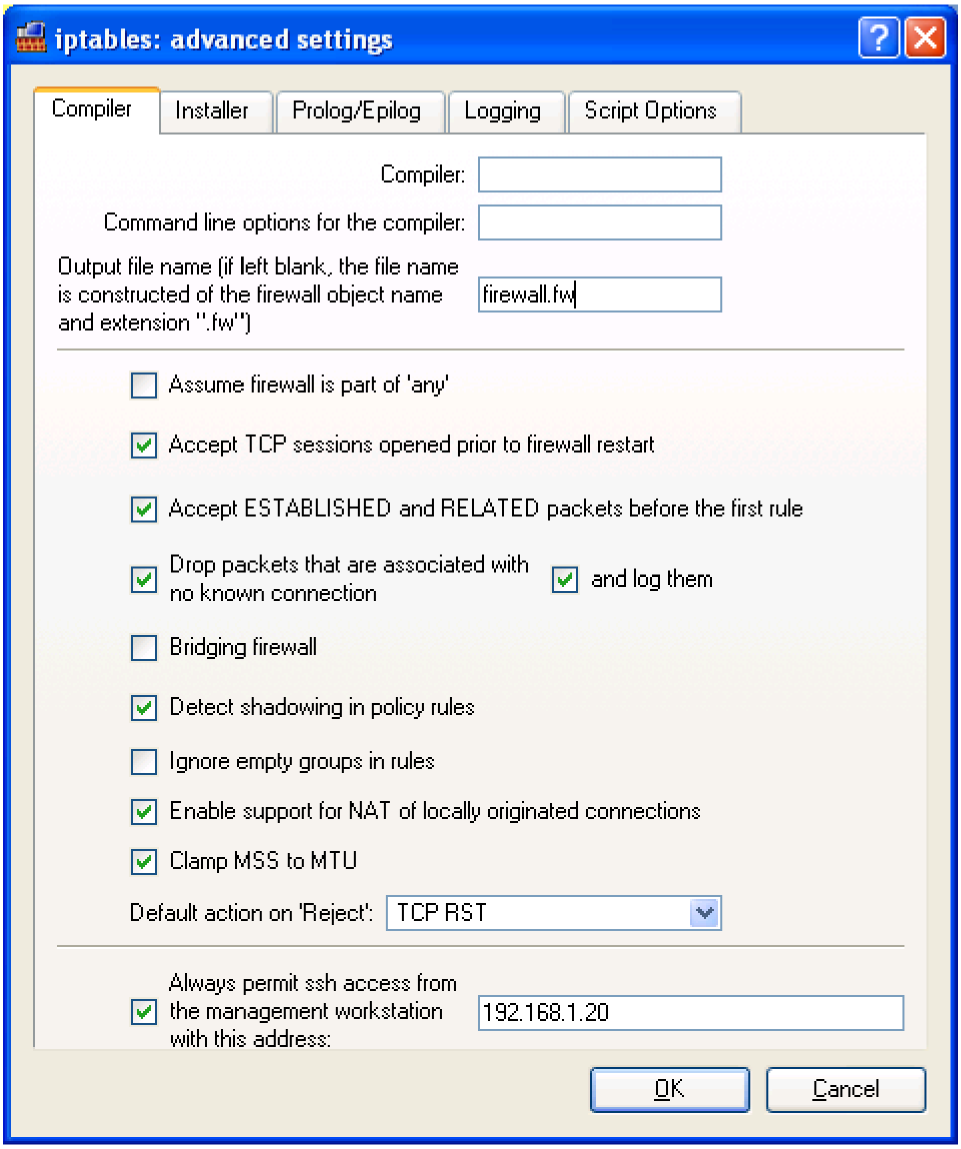
Set the platform to “iptables”
Set the version to “1.2.9 or latter”
Set the Host OS to “Linux 2.4/2.6”

There is an option under the “Host OS” setting for “Linksys/Sveasoft”, do not select this option as it will install the firewall rules in NVRAM. This can be a problem latter if the rules set is large and exhausts the NVRAM available.

The only disadvantage I have found to selecting the “Host OS” to “Linux 2.4/2.6”, is the default location of the modules. This means that you cannot have firewall builder automatically install modules, as small issue, as modules can be loaded else ware in openWrt

Firewall settings

Select “Firewall settings and ensure that the following parameters are set something similar to the following screen shots



Compiler:

Command line options for the compiler:

Output file name (if left blank, the file name is constructed of the firewall object name and extension ".fw"): firewall.fw

□ Assume firewall is part of 'any'

☑ Accept TCP sessions opened prior to firewall restart

☑ Accept ESTABLISHED and RELATED packets before the first rule

☑ Drop packets that are associated with no known connection ☑ and log them

□ Bridging firewall

☑ Detect shadowing in policy rules

□ Ignore empty groups in rules

☑ Enable support for NAT of locally originated connections

☑ Clamp MSS to MTU

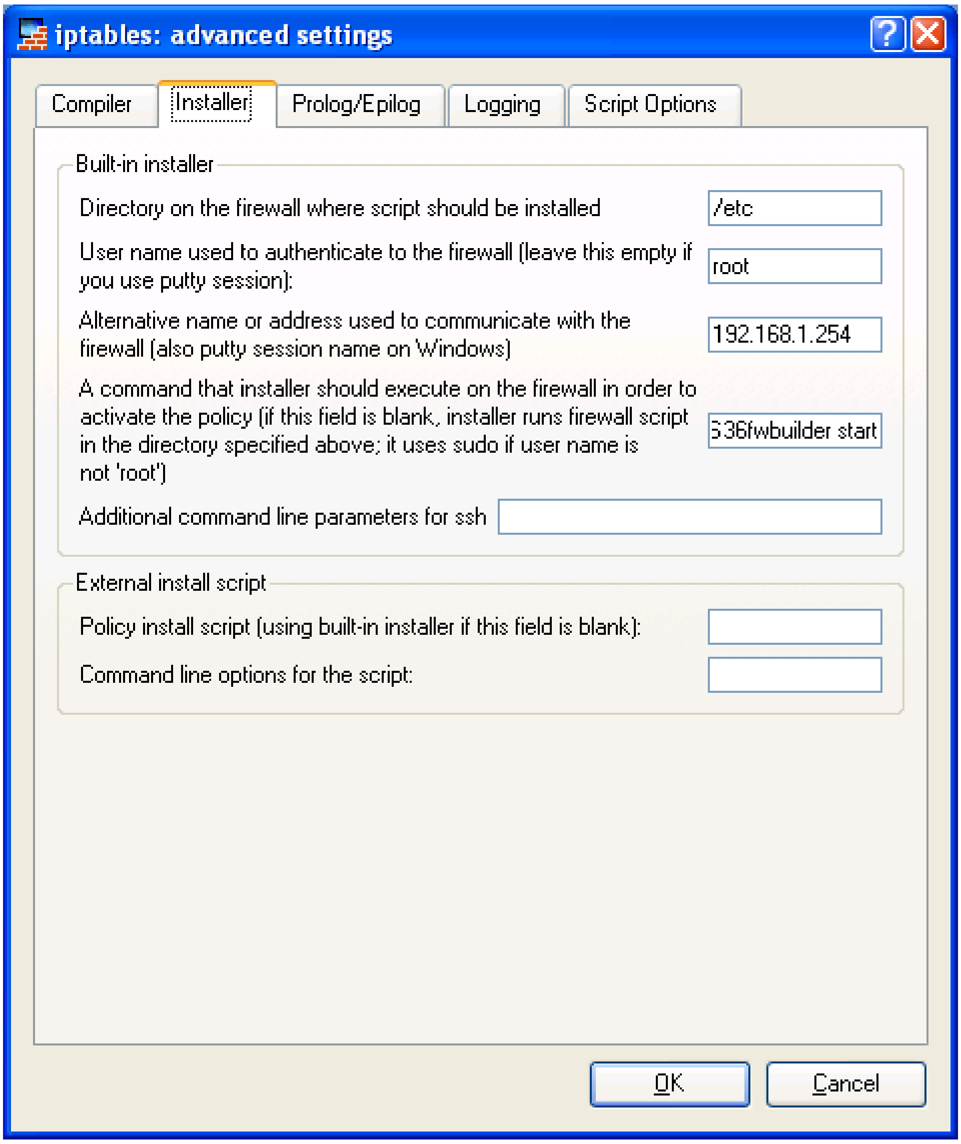
Default action on 'Reject': TCP RST

☑ Always permit ssh access from the management workstation with this address: 192.168.1.20

The most important field here is the name of the firewall script. This should be the same as you used earlier when setting up the router.

It is also a good idea to nominate an IP address that is always permitted to ssh to the router. This ensures that if you make a mess of the firewall rules that you will still be able to access the router and issue "/etc/init.d/S45firewall stop" to disable the firewall.

The rest of the fields should be specified as per your requirements.



Built-in installer

Directory on the firewall where script should be installed: /etc

User name used to authenticate to the firewall (leave this empty if you use putty session): root

Alternative name or address used to communicate with the firewall (also putty session name on Windows): 192.168.1.254

A command that installer should execute on the firewall in order to activate the policy (if this field is blank, installer runs firewall script in the directory specified above; it uses sudo if user name is not 'root'): S36fwbuilder start

Additional command line parameters for ssh:

External install script

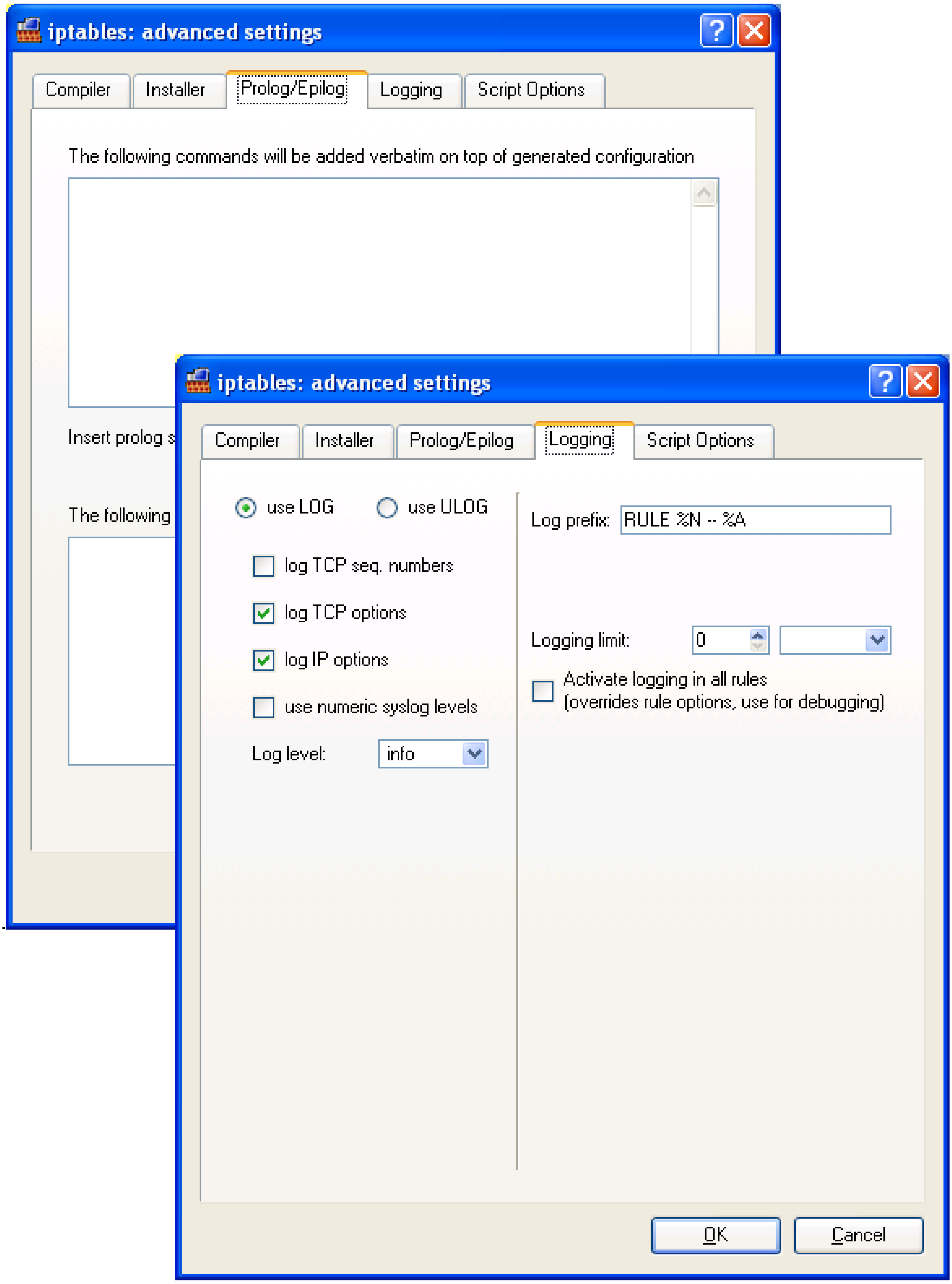
Policy install script (using built-in installer if this field is blank):

Command line options for the script:

NOTE: Not all the fields in this screen shot should be filled as shown.

(1) Use the IP address for your router

(2) The full string for the installer command is: "**/etc/init.d/S36fwbuilder start**"



The following commands will be added verbatim on top of generated configuration

Insert prolog s

The following

iptables: advanced settings

● use LOG ○ use ULOG

□ log TCP seq. numbers

☑ log TCP options

☑ log IP options

□ use numeric syslog levels

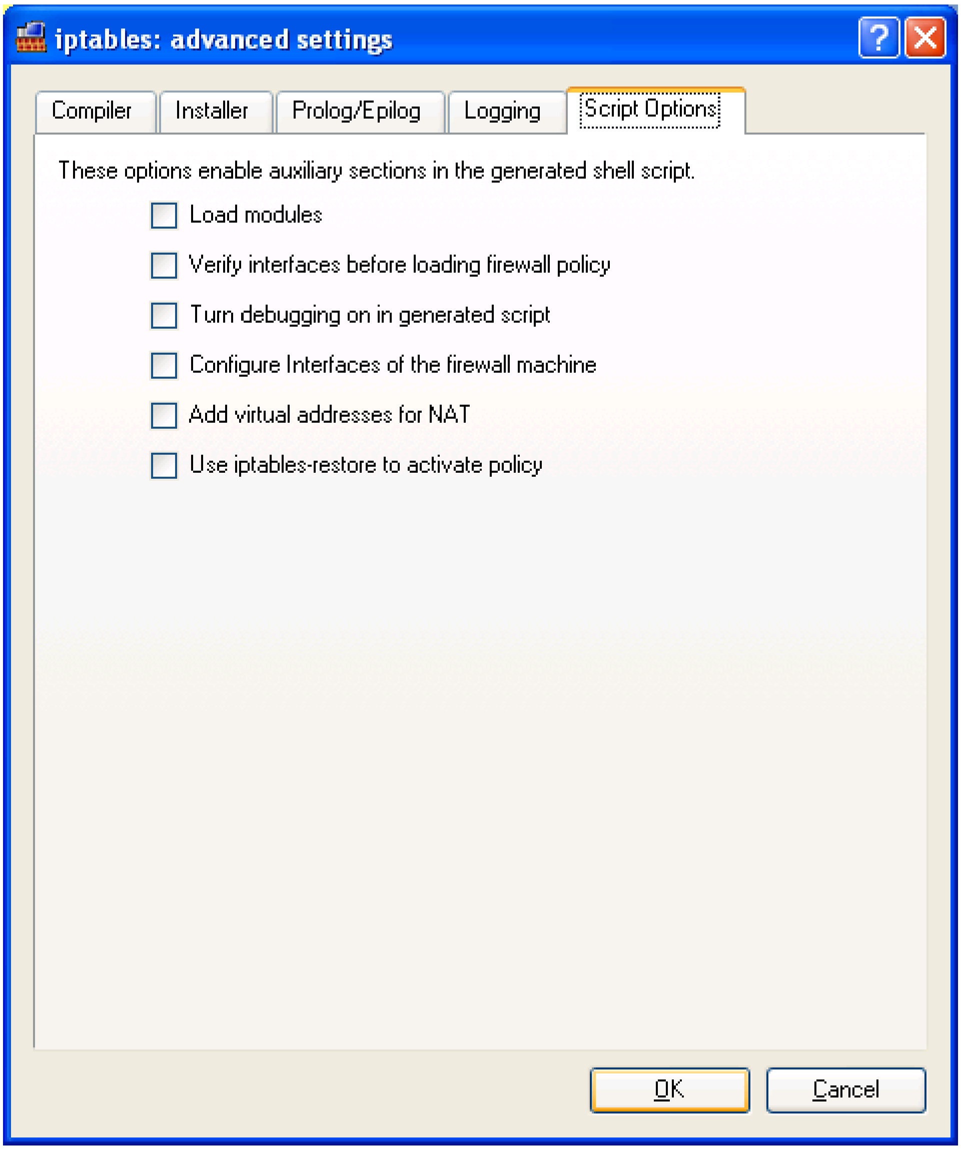
Log level: info

Log prefix: RULE %N -- %A

Logging limit: 0

□ Activate logging in all rules (overrides rule options, use for debugging)

The settings on these screen are not really important. You may want to customise you logging parameters to meet you needs.



These options enable auxiliary sections in the generated shell script.

- □ Load modules
- □ Verify interfaces before loading firewall policy
- □ Turn debugging on in generated script
- □ Configure Interfaces of the firewall machine
- □ Add virtual addresses for NAT
- □ Use iptables-restore to activate policy

This is the screen that matters the MOST….Don't select any of the parameters with the exception of enabling debugging, and "iptables-restore"

Loading Modules

If load modules is selected, fwbuilder will include the following code in the generated script.

```
      MODULE_DIR="/lib/modules/`uname -r`/kernel/net/ipv4/netfilter/"
      MODULES=`(cd $MODULE_DIR; ls *_conntrack_* *_nat_* | sed -n -e
's/\.ko$//p' -e 's/\.o$//p' -e 's/\.ko\.gz$//p' -e 's/\.o\.gz$//p')`
      for module in $MODULES; do
        if $LSMOD | grep ${module} >/dev/null; then continue; fi
        $MODPROBE ${module} ||  exit 1
      done
```

The path in to the modules directory is not combatable with openWrt. (in openWrt all modules are locate din the directory /lib/modules/2.4.30) So the fwbuilder script will fail. You could create a the extra directories and create a sym-link back to the openWrt directory at "/lib/modules/2.4.30/" however, this would mean that the script will load ALL the modules in the directory. Better to manually load the ones you need in either the start-up scripts or in /etc/modules

Verify Interfaces

This will check if all the interfaces are available before loading the firewall rules. If you have dynamic interfaces such as “ppp0” used for pppoe. They will not be present when the interface is not “up”, and as a result you will not be able to load the firewall rules set, until they are.

Debugging

This adds the “-x” option to the fwbuilder script. Can be useful at times. But rather than recompile the rule set, and copy to the router, I would log on to the router and issue the command:

```
sh -x /etc/firewall.fw
```

Configure Interfaces

The in interfaces are already configured. Doing so again will make a real mess

Virtual address for NAT

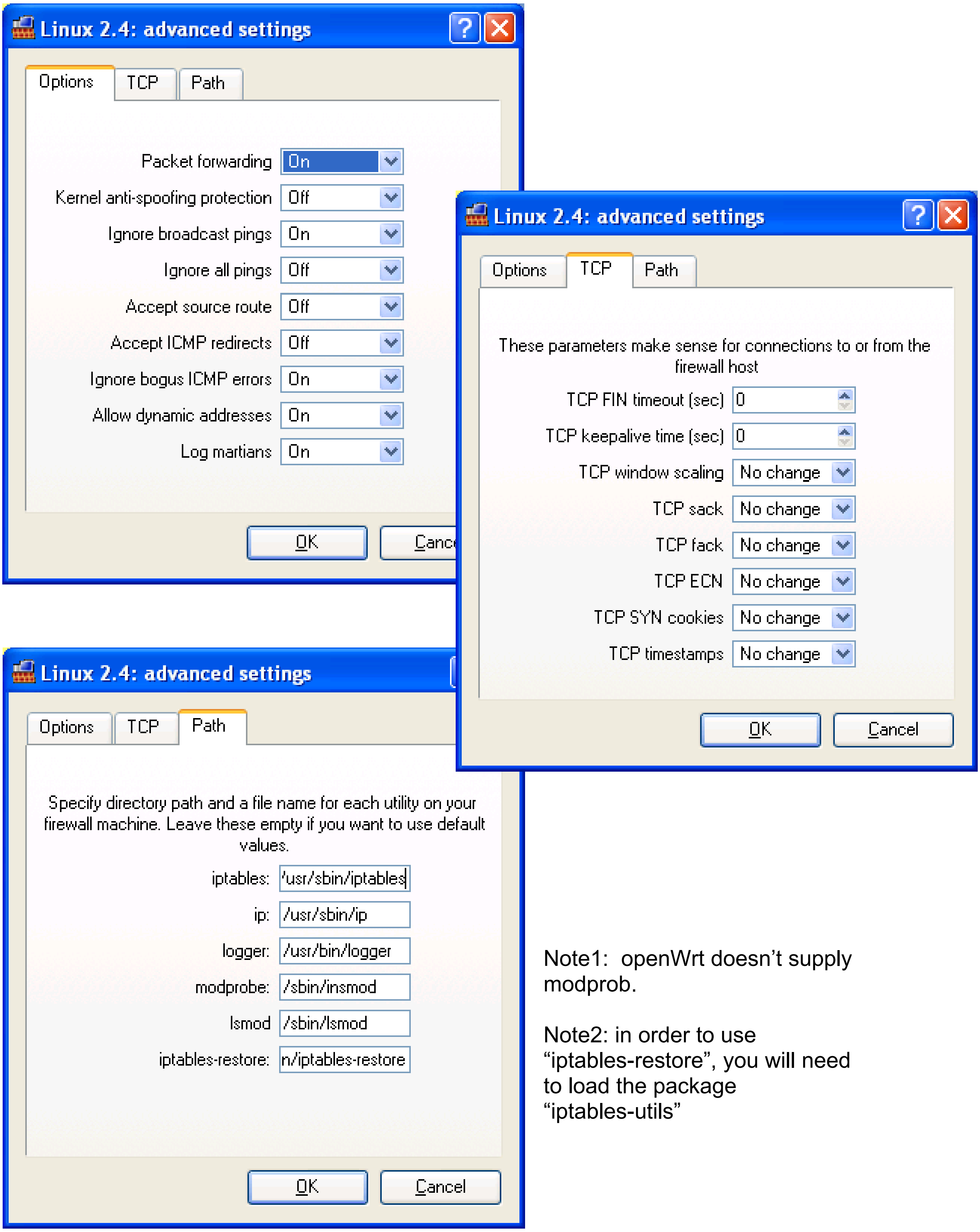
If you have multiple external addresses that you are NATing on your router, this could be useful. I honestly have not tried it yet. May be usefull where you need the NAT ip address on the external interface. For all the NAT that I have done, I have not needed the NAT IP address on the firewall external interface. The only reason that I can think that this would be required is if the external IP address of the firewall was in the same subnet as the NAT address.

Use iptables restore

The use of “iptables restore” makes the rules “much faster to load.
Earlier version of openwrt had issues with the use of “iptables restore”; however RC4+ appears to be fine. This is particularly useful if you have a large set of rules, which can take quite a while if this feature is not used

Host OS settings

The following screens show the Host OS settings that I have used



Packet forwarding: On
Kernel anti-spoofing protection: Off
Ignore broadcast pings: On
Ignore all pings: Off
Accept source route: Off
Accept ICMP redirects: Off
Ignore bogus ICMP errors: On
Allow dynamic addresses: On
Log martians: On

These parameters make sense for connections to or from the firewall host

TCP FIN timeout (sec): 0
TCP keepalive time (sec): 0
TCP window scaling: No change
TCP sack: No change
TCP fack: No change
TCP ECN: No change
TCP SYN cookies: No change
TCP timestamps: No change

Specify directory path and a file name for each utility on your firewall machine. Leave these empty if you want to use default values.

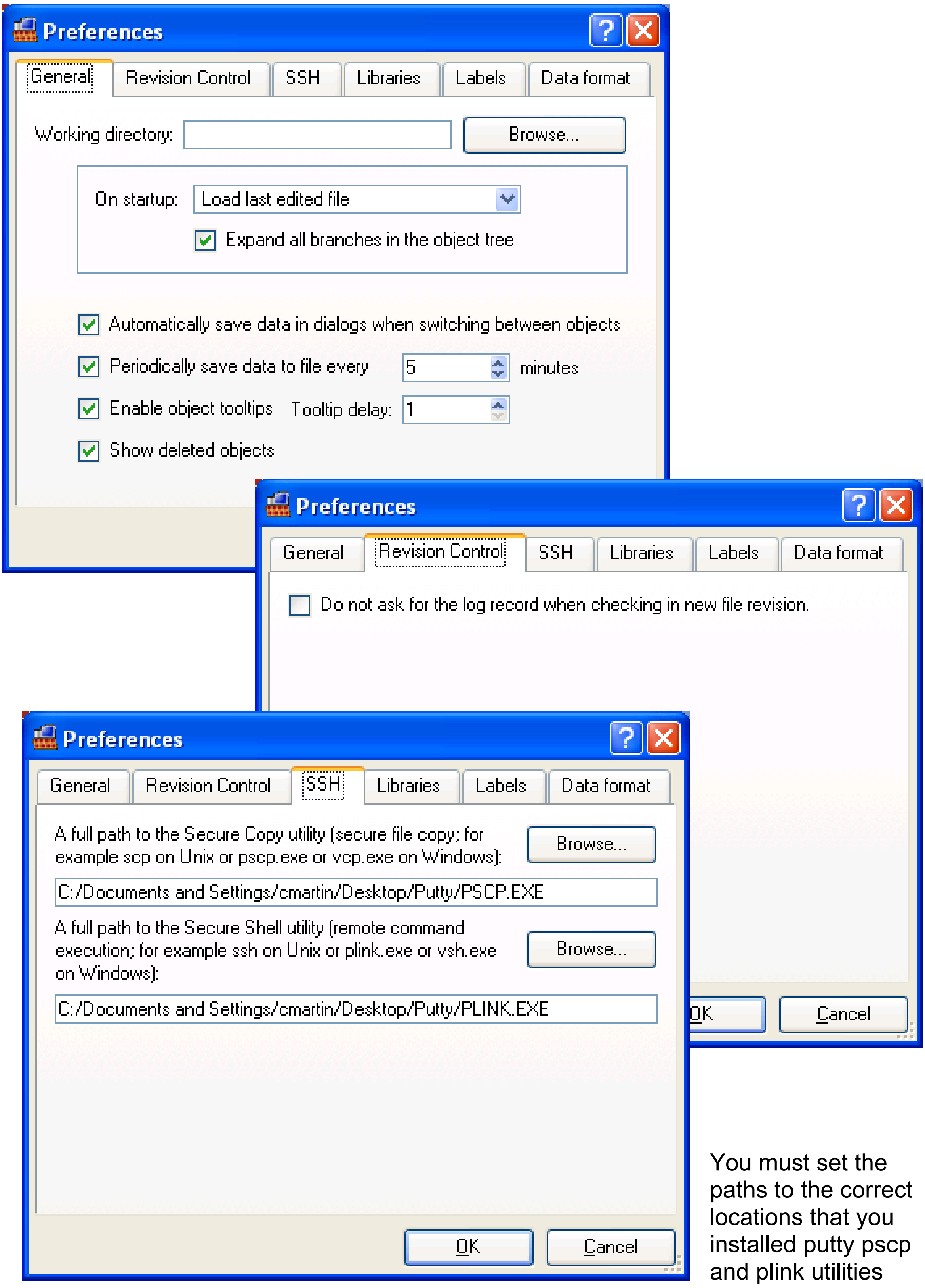
iptables: /usr/sbin/iptables
ip: /usr/sbin/ip
logger: /usr/bin/logger
modprobe: /sbin/insmod
lsmod: /sbin/lsmod
iptables-restore: n/iptables-restore

Note1: openWrt doesn’t supply modprob.

Note2: in order to use “iptables-restore”, you will need to load the package “iptables-utils”

Preference Settings

The preference settings are found under
The following screens show the Host OS settings that I have used



Working directory:

On startup: Load last edited file

☑ Expand all branches in the object tree

☑ Automatically save data in dialogs when switching between objects

☑ Periodically save data to file every 5 minutes

☑ Enable object tooltips Tooltip delay: 1

☑ Show deleted objects

□ Do not ask for the log record when checking in new file revision.

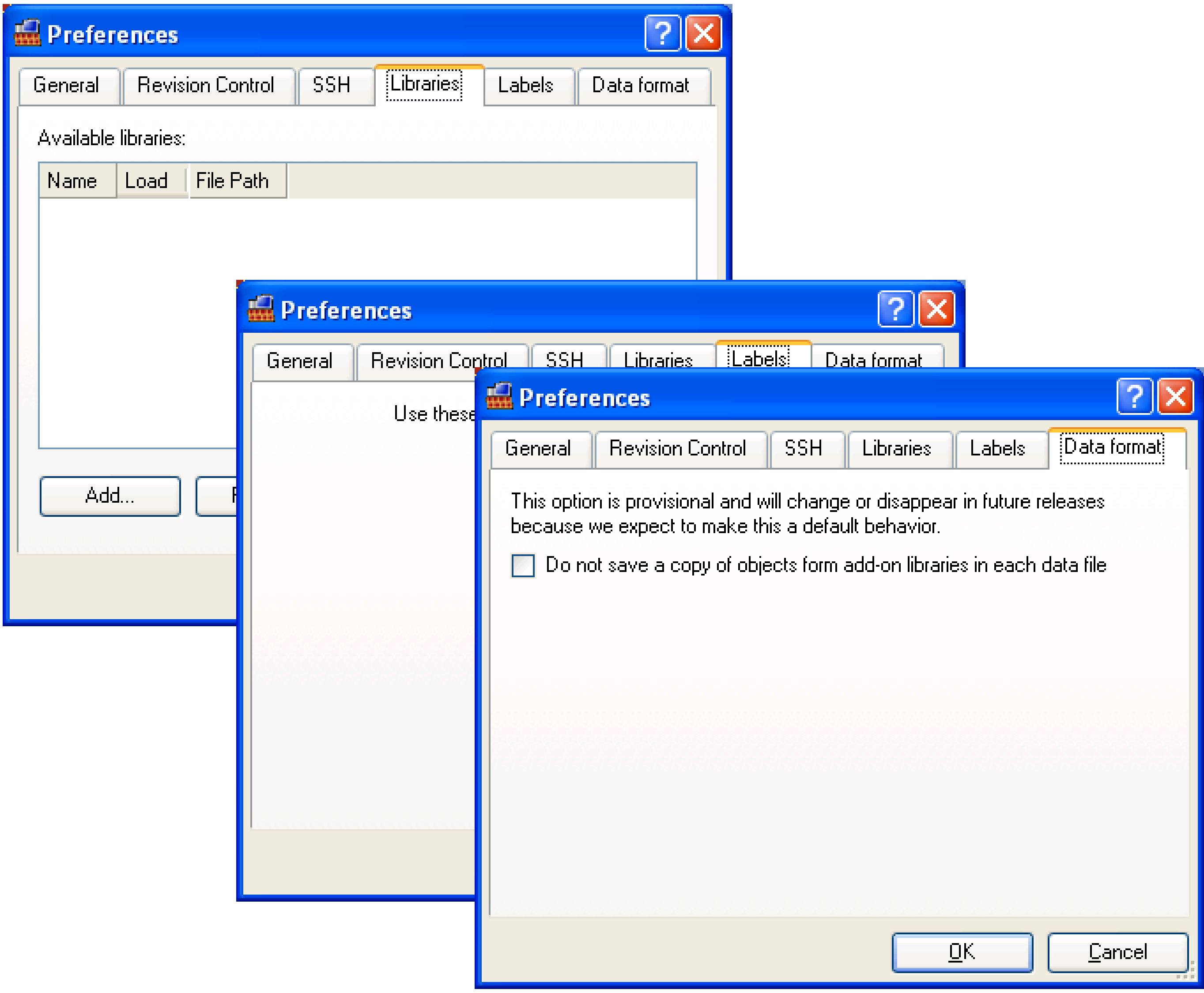
A full path to the Secure Copy utility (secure file copy; for example scp on Unix or pscp.exe or vcp.exe on Windows):

C:/Documents and Settings/cmartin/Desktop/Putty/PSCP.EXE

A full path to the Secure Shell utility (remote command execution; for example ssh on Unix or plink.exe or vsh.exe on Windows):

C:/Documents and Settings/cmartin/Desktop/Putty/PLINK.EXE

You must set the paths to the correct locations that you installed putty pscp and plink utilities



Available libraries:

This option is provisional and will change or disappear in future releases because we expect to make this a default behavior.

□ Do not save a copy of objects form add-on libraries in each data file

Utilities

“dyndnsupd” - DynDNS update utility

Create the script “**/etc/bin/dydnsupd**” with the following
chroot +x /etc/bin/dyndnsupd

This script is to be called from:
(1) /etc/hotplug/iface/ when the “wan” interface is brought up and an IP address is allocated:
(2) from cron to ensure that we make regular updates to keep your dyndns account active.

The script has the following features
(1) Emails errors to a nominated email account
(2) Logs to syslog
(3) Is run from cron every 15 minutes to check if the IP address has changed, and forces an IP address update every week to ensure that and account is remains active

This script also has a dependency on the mini-sendmail package.
Documentation is provided in the Appendix

The following the dyndnsupd script:
Make sure that it is executable

chown +x /etc/bin/dyndnsupd

```
#!/bin/sh

LOGGING_TAG="dyndns agent v1:"
LOGGER=/usr/bin/logger
ACCOUNT=userid
PASS=password
HOSTNAME=host.dyndns.org
FORCE_UPDATE_INTERVAL=604800
# SYSTEM=(dyndns | static | custom)
SYSTEM=dyndns
USER_AGENT="openwrt/1.0"
SMTP_SMARTHOST=”smtp@your.domain”
EMAIL_ADDR="username@your.domain"
RESULT_FILE=/tmp/dyndns.result

WAN_IFNAME=$(nvram get wan_ifname)
WAN_IFNAME=${WAN_IFNAME:-vlan1}


get_ip_from_wan_interface ()
{
        # attempt to get the current IP address for the WAN interface
        WAN_IP=$(ifconfig $WAN_IFNAME | grep "inet addr" | awk '{print
$2}' | awk -F ':' '{print $2}')

        # check that we got an IP address
        if [ "$WAN_IP" = "" ]
        then
```

```
                $LOGGER -t ${LOGGING_TAG} -- "ifconfig: couldn't get
an IP address for $WAN_IFNAME."
                exit 1
        fi

        echo $WAN_IP
        return 0
}

get_ip_from_dns ()
{
        # Get the IP address from DynDns
        DNS_IP=$(nslookup $HOSTNAME | sed s/[^0-9.]//g | tail -n1)

        # check that we got an IP address
        if [ "$DNS_IP" = "" ]
        then
                $LOGGER -t ${LOGGING_TAG} -- "nslookup: couldn't get
an IP address for $HOSTNAME."
                exit 1
        fi

        echo $DNS_IP
        return 0
}

get_last_update ()
{
        # see if we have done an update before
        [ -f $RESULT_FILE ] && LAST_UPDATE=$(date +%s -r $RESULT_FILE)
        [ -s $RESULT_FILE ] && read LAST_STATUS LAST_IP < $RESULT_FILE

        # make sure the values are ok
        LAST_STATUS=${LAST_STATUS:-"none"}
        LAST_IP=${LAST_IP:-$(get_ip_from_dns)}
        LAST_UPDATE=${LAST_UPDATE:-0}

        return 0
}

read_update_results ()
{
        # see if we have done an update before
        [ -s ${RESULT_FILE}.update ] && read UPDATE_STAUS UPDATE_INFO
< ${RESULT_FILE}.update

        # make sure the values are ok
        UPDATE_STATUS=${UPDATE_STAUS:-"none"}
        UPDATE_INFO=${UPDATE_INFO:-"undefined"}

        return 0
}


do_dyndns_update ()
{
        # NOTE: BusyBox (wget) doesnt have SSL(https) support so we
have to use HTTP in the clear

        wget -q -O ${RESULT_FILE}.update --header "User-Agent:
$USER_AGENT $EMAIL_ADDR" "http://${ACCOUNT}:$
```

```
{PASS}@members.dyndns.org/nic/update?system=${SYSTEM}&hostname=$
{HOSTNAME}&myip=${WAN_IP}&wildcard=OFF&offline=NO"

        read_update_results

        # check the status
        case $UPDATE_STATUS in

        "good" | "nochg" )
                # update performed OK
                $LOGGER -t ${LOGGING_TAG} -- "Successfull update,
status=${UPDATE_STATUS}, IPaddress=${UPDATE_INFO}"
                mv ${RESULT_FILE}.update ${RESULT_FILE}
                exit 0
                ;;

        "badsys" )
                ERROR_MSG="The \$SYSTEM parameter in \$0 is not valid.
Valid values are \"dyndns\", \"statdns\" and \"custom\"."
                ;;

        "badagent" )
                ERROR_MSG="Updates have been blocked for either: (1)
not following the update guideliens or (2) no User-Agent specified."
                ;;

        "badauth" )
                ERROR_MSG="The username and/or password specified are
incorrect."
                ;;

        "!donator" )
                ERROR_MSG="An option available only to credited users
(such as offline URL) was specified, but the user is not a credited
user."
                ;;

        "notfqdn" )
                ERROR_MSG="The hostname specified is not a
fully-qualified domain name (not in the form hostname.dyndns.org or
domain.com etc)."
                ;;

        "nohost" )
                ERROR_MSG="The hostname specified does not exist (or
is not in the service specified in the system parameter)."
                ;;

        "!yours" )
                ERROR_MSG="The hostname specified exists, but not
under the username specified."
                ;;

        "abuse" )
                ERROR_MSG="The hostname specified is blocked for
update abuse."
                ;;

        "numhost" )
                ERROR_MSG="Too many or too few hosts found."
                ;;
        "dnserr" )
                ERROR_MSG="DNS error encountered. ERROR_NUM=$
{UPDATE_INFO}. Please call DynDNS to resolve"
                ;;

        "911" )
                ERROR_MSG="There is a serious problem on our side,
such as a database or DNS server failure. Please should stop updating
until the service is back up."
                ;;

        * )
                ERROR_MSG="An undefined error has occured.  This is
either due to, (1) a dyndns error condition not caught by this script,
or (2) a possible network problem."
                ;;
        esac

        # if we get here we have an error
        STATUS_MSG="Update ERROR, status=${UPDATE_STATUS}, IPaddress=$
{WAN_IP}, HOSTNAME=${HOSTNAME}, Account=${ACCOUNT}"

        # log it
        $LOGGER -t ${LOGGING_TAG} -- $STATUS_MSG
        $LOGGER -t ${LOGGING_TAG} -- $ERROR_MSG

        # mail it
        (echo "To: <${EMAIL_ADDR}>"
        echo "From: <${EMAIL_ADDR}> OpenWrt FireWall"
        echo "Subject: OpenWrt - DynDns Update Error"
        echo
        echo $STATUS_MSG
        echo $ERROR_MSG
        echo "<eom>"
        ) |  mini_sendmail -f"${EMAIL_ADDR}" -s"${SMTP_SMARTHOST}" -t

        # get rid of the last result file...  it was a failure
        [ -f ${RESULT_FILE}.update ] && rm ${RESULT_FILE}.update

        # bye bye
        exit 0
}

#############################
# Start
#############################

# get all the variables
get_ip_from_wan_interface
get_last_update

# if the IP address are not the same we must do an update
[ "$WAN_IP" != "$LAST_IP" ] && do_dyndns_update && exit 0

# if we have reciently done an update exit
[ $(expr $(date +%s) - $LAST_UPDATE) -gt $FORCE_UPDATE_INTERVAL ] &&
do_dyndns_update && exit 0

# if we get to here..  we dont need to do anything
exit 0
```

“spinlock” – ensure that only one copy of a script runs at once

The following is the script “**/etc/bin/spinlock**”:
Make sure it is executable by issuing the following command

chown +x /etc/bin/spinlock

This script is to be called from (1) /etc/ppp/ip-up when the and IP address is allocated, and from cron to ensure that we make regular updates to keep pour dyndns account active.

```
#!/bin/sh

#########################
# Tunable Parameters
#########################
# How long to wait for a lock to clear in seconds
SPINWAIT=300


kill_locked_process ()
{
# Arguments passed
# $1 = process to kill PID

        # This will attempt to kill a locked process
        KILL_PID=$1
        if [ "$KILL_PID" != "" ] && [ $KILL_PID -gt 100 ]
        then
                # log what we are doing
                ps | grep $KILL_PID | while read LINE
                do
                        logger -p info "$0: Time out waiting for lock, 
terminating process: $LINE"
                done

                # terminate the process
                kill $KILL_PID 2>/dev/null

                # wait a bit for process to die
                sleep 5

                # try again with extreme prejudice
                kill -15 $KILL_PID 2>/dev/null
        fi
}


wait_for_lock()
{
# $1=Patience interval

        # PATIENCE is how long to wait for the lock to free up
        PATIENCE=$1
        PATIENCE=${PATIENCE:-SPINWAIT}

        # start time
        START_PID=0
        START_TIME=$(date +%s)

        # check if there is a semaphore/lock file
        # does file exist and length gtr 0
        while [ -s $SPINLOCK_FILE ]
        do
                # check if the process is running
                read LOCK_PID < $SPINLOCK_FILE
                [ "$LOCK_PID" != "" ] && kill -0 $LOCK_PID 2>/dev/null
|| break

                # check that no-one else got the lock while we where
sleeping
                [ "$START_PID" = 0 ] && START_PID=$LOCK_PID
                [ "$START_PID" != "$LOCK_PID" ] &&
(START_PID=$LOCK_PID; START_TIME=$(date +%s))

                # check if we have been waiting tooooo long (ie: run
out of patience)
                [ $(expr $(date +%s) - $START_TIME) -gt $PATIENCE ]
&& kill -9 $LOCK_PID

                # wait a bit
                sleep 5
        done

        #  attempt to acquire the lock
        echo "$$" > $SPINLOCK_FILE
}

clear_lock()
{
        # after program is complete, clear the lock
        > $SPINLOCK_FILE
}

verify_lock()
{
        # verify that we got  the lock
        read LOCK_PID < $SPINLOCK_FILE
        case "$LOCK_PID" in
        $$)
                # we have the lock
                trap clear_lock 0 9 15
                return 0;
                ;;
        *)
                # Not our lock
                return 1;
        esac
}

usage()
{
        # display usage information
        ME=$(basename $0)
        echo "usage: $ME <command>"
        echo "multiple commands should be placed in a script"
        exit 1
}
```

```
#######################################
# START
#######################################

# check that this is called with parameters
[ $# -ne "1" ] && usage

# work out required Commands & Parameters
SPINLOCK_FILE=/tmp/$(basename $1).pid

# Spin waiting for the lock
while :
do
        wait_for_lock
        if verify_lock
        then
                # Run the command
                "$@"

                # remove the lock
                clear_lock

                # bye
                exit
        fi
done
```

“pkill” – process kill

This script is only required if you are using Firewall Builder. The script generated by fwbuilder, depending on the options selected will expect to find the “pkill”. This command is not provided in openWrt, however the killall command is in RC4+

I have kept an older script, in comments just in case they take the “killall” command away in the future

Create the script “**/etc/bin/pkill**”:
Make sure it is exacutable:

chown +x /etc/bin/pkill

```
#!/bin/sh

exec killall $*

# NOTE..  The following commented lines have been left just in case
# the opernWrt guys take killall out of busybox
#
# case $# in
# 1)
#       SIG="-15"
#       STRING="$1"
#       ;;
# 2)
#       SIG="$1"
#       STRING="$2"
#.      ;;
# *)
#       echo "usage: $0 [-signal] string"
#       exit 1
#       ;;
# esac
#
# ps | grep $STRING | grep -v grep | grep -v $0 | while read pid
restOfLine
# do
#         kill $SIG $pid 2>/dev/null
# done
```

Appendix

Crontab file format

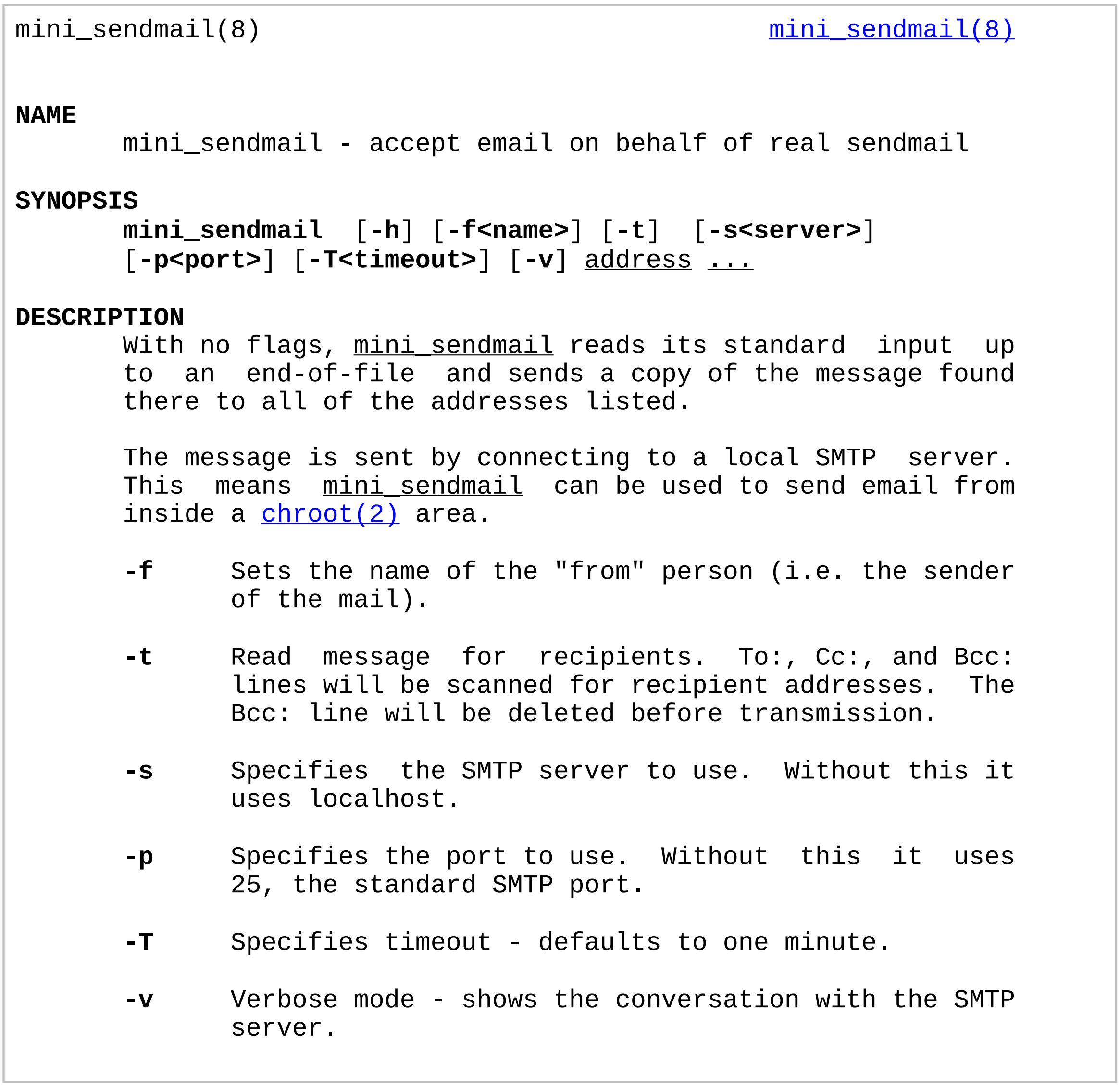
The format for the crontab file is:

```
field         allowed values
-----         --------------
minute        0-59
hour          0-23
day of month  1-31
month         1-12 (or names, see below)
day of week   0-7 (0 or 7 is Sun, or use names)

Note:
      (1)  "*" is a shortcut for "first-to-last"
      (2)  Ranges(-) and lists(,) are allowed, and can be mixed
              eg: "1,2,5,9", "0-4,8-12".
      (3)  Values can be stepped
              eg: "0-23/2"


Example crontab file
```

```
# check the IP address every hour, on the hour
0 * * * * /usr/bin/checkmyip
# Get the time each day @ midnight
0 0 * * * /usr/sbin/rdate -s india.colorado.edu
# reboot at every night at 5am in the morning
0 5 * * * /sbin/reboot
```

“mini_sendmail” command Reference

The following is documentation on “mini_sendmail”



```
mini_sendmail(8)                                   mini_sendmail(8)


NAME
       mini_sendmail - accept email on behalf of real sendmail

SYNOPSIS
       mini_sendmail  [-h] [-f<name>] [-t]  [-s<server>]
       [-p<port>] [-T<timeout>] [-v] address ...

DESCRIPTION
       With no flags, mini_sendmail reads its standard  input  up
       to  an  end-of-file  and sends a copy of the message found
       there to all of the addresses listed.

       The message is sent by connecting to a local SMTP  server.
       This  means  mini_sendmail  can be used to send email from
       inside a chroot(2) area.

       -f     Sets the name of the "from" person (i.e. the sender
              of the mail).

       -t     Read  message  for  recipients.  To:, Cc:, and Bcc:
              lines will be scanned for recipient addresses.  The
              Bcc: line will be deleted before transmission.

       -s     Specifies  the SMTP server to use.  Without this it
              uses localhost.

       -p     Specifies the port to use.  Without  this  it  uses
              25, the standard SMTP port.

       -T     Specifies timeout - defaults to one minute.

       -v     Verbose mode - shows the conversation with the SMTP
              server.
```

Adding a Secondary IP address to an Interface

A secondary, or other address can be added to an interface to do this use the command
Incriment the interface index each time you add a new IP address

```
ifconfig <ifname>:1 <some.secondary.ip.address>
ifconfig <ifname>:2 <some.secondary.ip.address>
```

You may also need to add appropriate routes as well