OpenWrtDocs

Contents

1. NVRAM

NVRAM stands for Non-Volatile RAM, in this case the last 64K of the flash chip used to store various configuration information in a *name=value* format.

Command	Description
nvram show | sort | less	Display everything in NVRAM
nvram get boot_wait	Get a specific variable
nvram set boot_wait=on	Set a value
nvram set lan_ifnames="vlan0 vlan1 vlan2"	set multiple values to one param
nvram unset foo	Delete a variable
nvram commit	Write changes to the flash chip (otherwise only stored in RAM)

A complete list of nvram options can be found at OpenWrtNVRAM.

2. Network configuration

Quick overview of the router architecture:

The WRT54G is made up of an Ethernet switch, a wireless access point and a router chip that connects them together.

Diagrams of the internal switch architectures can be found via the following table

Device & Version	
WRT54G v2/v3 & WRT54GS v1/v2	Switch diagram
WRT54G v4 & WRT54GS v3	Switch diagram

The names of the network interfaces will depend largely on what hardware OpenWrt is run on. A more detailed explanation of the networking internals is on the page OpenWrtDocs/NetworkInterfaces

Manufacturer	Model	Hardware version	LAN	WAN	WIFI	Comments
Linksys	WRT54G	v1.x	vlan2	vlan1	eth2	
Linksys	WRT54G	v2.x/v3.x/v4.0	vlan0	vlan1	eth1	
Linksys	WRT54GL	v1.0	vlan0	vlan1	eth1	
Linksys	WRT54GL	v1.1	vlan0	vlan1	eth1	LAN is ports 0-3, WAN is port 4
Linksys	WRT54GS	v1.x/v2.x/v3/v4	vlan0	vlan1	eth1	
Linksys	WRTSL54GS		eth0	eth1	eth2	
Linksys	WAP54G	v1.0	br0	N/A	eth1	Someone should double check this too
Linksys	WAP54G	v2.0	eth0	N/A	eth1	note[2]
Linksys	WRT300N	v1	eth0	eth1	eth2	
Asus	WL-300g		eth0	None	eth2	

Asus	WL-500g		eth0	eth1	eth2	
Asus	WL-500g Deluxe		vlan0	vlan1	eth1	note[1]
Asus	WL-500g Premium		vlan0	vlan1	eth2	note[1]
Asus	WL-HDD		eth1	N/A	eth2	No switch and no WAN port
Belkin	OpenWrtDocs/Hardware/Belkin/F5D7130	1010	eth0	eth1	eth2	By default, LAN is br0 bridging eth0 and eth2
Buffalo	WBR-G54		eth0	eth1	eth2	
Buffalo	WBR2-G54		vlan0	vlan1	eth1	note[1]
Buffalo	WBR2-G54S		vlan0	vlan1	eth1	note[1]
Buffalo	WHR-G54S		vlan0	vlan1	eth1	note[1]
Buffalo	WHR-G54S	SN:7407	br0	vlan1	eth1	SVN-2006-09-15
Buffalo	WLA-G54		eth0	N/A	eth2	No WAN port on this device
Buffalo	WZR-RS-G54		eth0	eth1	eth2	no vlan support (switch BCM5325A2KQM)
Buffalo	WHR3-G54		eth0	eth1	eth2	no vlan support (switch BCM5325A2KQM)
Dell	TrueMobile 2300		eth0	eth1	eth2	BCM5325MA2KQM switch
Motorola	WR850G	v3	vlan0	vlan1	eth1	note[1]
Microsoft	MN700	v.x	eth0	eth1	eth2	
Netgear	WGT-634U		vlan0	vlan1	ath0	note[1]
Siemens	SE505	v1	eth0	eth1	eth2	
Siemens	SE505	v2	vlan0	vlan1	eth1	note[1]

note[1]: This model uses a switch with vlan tagging; eth0 represents the connection from the router to the switch and the vlans ontop of eth0 will control which switch port(s) the packet is transmitted.

note[2]: Be careful: after flashing with OpenWRT, LAN stops working. Before flashing, set WAP54g to AP mode and after OpenWRT has been loaded, connect to the device through wireless and do: `nvram set lan_ifnames=eth0 eth1 ; nvram commit`

Please update to include other models.

NOTE: LAN and WIFI are bridged together in br0 by default, on some devices WAN can be eth1 and LAN eth0.

The basic network configuration is handled by a series of NVRAM variables:

NVRAM	**Description**
<name>_ifname	The name of the Linux interface the settings apply to
<name>_ifnames	Devices to be added to the bridge (only if the above is a bridge)
<name>_proto	The protocol which will be used to configure an IP
	static: Manual configuration (see below)
	dhcpclient: Perform a DHCP request (used to be just "dhcp")
	pppoe: Create a ppp tunnel
<name>_ipaddr	ip address (x.x.x.x)
<name>_netmask	netmask (x.x.x.x)
<name>_gateway	Default Gateway (x.x.x.x)
<name>_dns	DNS server (x.x.x.x)
<name>_hostname	hostname requested with dhcp
<name>_hwaddr	MAC address (aa:bb:cc:dd:ee:ff) if you want to use a different MAC of the ROM

Where <name> is either one of 'wl0', 'lan', or 'wan' for the wireless, local area network, or the wide area network respectively.

The command *ifup <name>* will configure the interface defined by <name>_ifname according to the above variables. As an example, the `/etc/init.d/S40network` script will automatically run the following commands at boot:

```
ifup lan
ifup wan
ifup wifi
```

The *ifup lan* command will bring up the interface specified by lan_ifname. Normally the lan_ifname is set to br0 which will cause it to create the bridge br0 and add the the interfaces from lan_ifnames to the bridge; lan_proto is usually static which means that br0 will have the IP address from lan_ipaddr, and so on for the rest of the variables listed above.

It's important to remember that it's the <name>_ifname that specifies the interfaces, the <name> component itself has almost no value. This means that if you changed lan_ifname to be the internet port, vlan1, then *ifup lan* would bring up the internet port, not the lan ports (despite using the command *ifup lan* and using the lan_ variables). Also, it means that you can create any <name> variables you want, foo_ifname, foo_proto ....

and they would be used by *ifup foo*.

The only <name> with any significance is **wan**, used by the /etc/init.d/S45firewall script. The firewall script will NAT traffic through the wan_ifname, blocking connections to wan_ifname.

Further information about the variables used can be found at OpenWrtNVRAM.

Don't forget to check the OpenWrtFaq for information about howto setup PPPoE etc.

Sample network configurations

For client mode configuration (rather than AP mode), see this page: ClientModeHowto. For further information on **DHCP** see this page dnsmasq

(**NOTE:** these examples use WRT54G v2.x/WRT54GS v1.x interface names)

The default network configuration (LAN + wireless bridged as 192.168.1.1/24, WAN as DHCP):

```
lan_ifname=br0
lan_ifnames="vlan0 eth1"
lan_proto=static
lan_ipaddr=192.168.1.1
lan_netmask=255.255.255.0
wan_ifname=vlan1
wan_proto=dhcp
```

If you just want to use OpenWrt as an access point you can avoid the WAN interface completely (LAN+wireless bridged as 192.168.1.25/24, routed through 192.168.1.1, WAN ignored):

```
lan_ifname=br0
lan_ifnames="vlan0 eth1"
lan_proto=static
lan_ipaddr=192.168.1.25
lan_netmask=255.255.255.0
lan_gateway=192.168.1.1
lan_dns=192.168.1.1
wan_proto=none
```

The above configuration also serves as a wireless to Ethernet bridge. For e.g. you can have a PC with a wlan card with a static IP address be switched (bridged) to an Ethernet LAN. Neither the IP address of the lan gateway, or the dhcp server on the LAN interface interferes with this bridged configuration.

You can also have the lan interface fetch its configuration via DHCP, but to do so, you'll have to comment out the line:

```
# linksys bug; remove when not using static configuration for lan
nvram set lan_proto="static"
```

in /etc/init.d/S05nvram (For RC5 and earlier the usual story about replacing the symlink with a copy of the file before editing applies, see Editing files at OpenWrtDocs/Using). After doing this, you need to set the appropriate nvram variable:

```
lan_proto=dhcp
```

To separate the LAN from the WIFI (LAN as 192.168.1.25/24, wireless as 192.168.2.25/24, WAN as DHCP, remove your WIFI interface (eth1 on v2/3 linksys routers) from the lan_ifnames variable):

```
lan_ifname=vlan0
lan_proto=static
lan_ipaddr=192.168.1.25
lan_netmask=255.255.255.0
wifi_ifname=eth1
wifi_proto=static
wifi_ipaddr=192.168.2.25
wifi_netmask=255.255.255.0
wan_ifname=vlan1
wan_proto=dhcp
lan_ifnames=vlan0
```

You MUST do this if you want to use ad-hoc mode, otherwise your throughput WILL suffer!

⚠ **Tip:** Don't forget to adjust packet filtering. For instance: `iptables -I forwarding_rule -j ACCEPT` enables packet forwarding (good for test, but insecure for production).

3. Ethernet switch configuration

Most of the routers supported by OpenWrt include a builtin switch; four lan ports and one wan port. What most people don't realize is that all of these ports are actually the same interface -- there is a single 10/100 Ethernet which is fed into a 6 port switch. 5 of the ports are external and make the lan and wan ports seen on the back of the router, and one port is internally wired to the router's Ethernet interface.

The separation of lan and wan comes from the use of VLANs. By grouping ports into VLANs, the switch can be broken up into smaller virtual switches, and by adding VLAN tags to packets, OpenWrt can control which virtual switch (which ports) the packet gets routed.

There are normally two VLANs, vlan0 and vlan1. For each VLAN, there are two nvram variables, vlan*ports and vlan*hwname. So, the variables for vlan0 might look like this:

```
vlan0ports="1 2 3 4 5*" (use ports 1-4 on the back, 5 is the WRT54G itself)
vlan0hwname=et0
```

(See switch diagrams in OpenWrtDocs/NetworkInterfaces)

The vlan0ports variable is a space-separated list of port numbers to be included in vlan0. Ports "1-4" on this router represent the lan ports on the back of the router, port 5 represents the connection between the switch itself and OpenWrt's Ethernet interface. Since port 5 is OpenWrt's only connection to the switch, it is tagged by default -- this means that the VLAN information is preserved so OpenWrt is able to tell if a packet came from vlan0 or vlan1. All other ports are untagged by default, meaning that the VLAN information is removed by the switch so the port can be used by devices that aren't VLAN aware.

The port numbers used in the vlan*ports may optionally include a character after the port number. If a port number is followed by a "t" then the port is tagged, a "u" means untagged.

A "*" means that this VLAN is the primary VLAN (PVID); if a port is used in multiple vlans, packets without any VLAN information will be given to the primary VLAN for that port.

The second variable, vlan0hwname is used by the network configuration program (the ifup scripts) to determine the parent interface. This should be set to "et0" meaning the interface matching et0macaddr. The reason it's labeled "et0" and not "eth0" is mostly due to vxworks -- it's a legacy issue and OpenWrt keeps the "et0" name to be compatible with the existing settings.

As of RC4, the switch is programmed and controlled by a set of switch modules (switch-core and switch-robo or switch-adm, depending on your hardware). These switch modules will create a /proc/switch/eth0, showing the current settings for the switch.
The /proc/switch/eth0/vlan/0/ports is used the exact same way as the vlan0ports nvram variable, allowing you to change the switch settings in realtime.

Sample configurations (unless otherwise specified, vlan variables not shown are assumed to be unset)

Default:

```
vlan0ports="1 2 3 4 5*"
vlan0hwname=et0
vlan1ports="0 5"
vlan1hwname=et0
```

All ports lan (vlan0):

```
vlan0ports="0 1 2 3 4 5*"
vlan0hwname=et0
```

LAN (vlan0), WAN (vlan1), DMZ (vlan2):

```
vlan0ports="1 2 5*"
vlan0hwname=et0
vlan1ports="0 5"
vlan1hwname=et0
vlan2ports="3 4 5"
vlan2hwname=et0
```

It's a good idea when choosing a vlan layout to keep port 1 in vlan0. At least the WRT54GS v1.0 will not accept new firmware via TFTP if port 1 is in another VLAN.

4. Wireless configuration

4.1. Basic settings

NVRAM variable	Description
wl0_mode	**ap** = Access Point (master mode), **sta** = Routing client mode, **wet** = Bridged client mode
wl0_ssid	ESSID
wl0_infra	**0** = Ad Hoc mode, **1** = normal AP/Client mode
wl0_closed	**0** = Broadcast ESSID, **1** Hide ESSID
wl0_channel	1 / 2 / 3 /.../ 11 channel

See OpenWrtNVRAM for more NVRAM settings.

4.2. MAC filter

NVRAM variable	Description
wl0_macmode	(disabled/allow/deny) used to (allow/deny) mac addresses listed in wl0_maclist
wl0_maclist	List of space-separated mac addresses to allow/deny according to wl0_macmode. Addresses should be entered with colons, e.g.: "00:02:2D:08:E2:1D 00:03:3E:05:E1:1B". note that if you have more than one mac use quotes or only the first will be recognized.

After changes run /sbin/wifi to activate them.

4.3. WEP encryption

NVRAM variable	Description
wl0_wep	**disabled** = disabled WEP, **enabled** = enable WEP

wl0_key	**1** .. **4** = Select WEP key to use
wl0_key [1..4]	WEP key in hexadecimal format (allowed hex chars are 0-9a-f). **Example:** nvram set wl0_key1=0D77F08849E4B1D839C9489A48
wl0_auth	**1** (shared key) / **0** (open); the 'shared key' option is not recommended as it allows an intruder to exploit a fundamental security flaw in WEP (WPA was introduced as the better system; see below). The 'open' setting will allow association but will make it an intruder more difficult to find the encryption key, needed for traffic.

Avoid using WEP keys with 00 at the end, otherwise the driver won't be able to detect the key length correctly. A 128-bit WEP key must be 26 hex digits long ; string key format is also supported : **nvram set wl0_key1='s:my string key'**

Setting up WPA will override any WEP settings.

4.4. WPA encryption

For enabling WPA, you need to install the nas package. When you enable or disable WPA settings, you should make sure that the NVRAM variable **wl0_auth_mode** is unset, because it is obsolete.

YOU HAVE TO INSTALL THE NAS PACKAGE (`ipkg install nas`)

More information is on OpenWrtDocs/nas.

See OpenWrtDocs/Wpa2Enterprise for a detailed setup using Freeradius for user authentication.

NVRAM variable	Description
wl0_akm	**open** = No WPA; Note: OpenWRT v0.9 uses the value **none**
	psk = WPA Personal/PSK (Preshared Key)
	wpa = WPA with a RADIUS server
	psk2 = WPA2 PSK
	wpa2 = WPA2 with RADIUS
	"psk psk2" or **"wpa wpa2"** = support both WPA and WPA2 **Note:** Do not use this value when wl0_mode=sta because supplicant mode does not seem to auto-negotiate. You must select one protocol which the access point supports (refer to the AP's specs)
wl0_crypto	**tkip** = RC4 encryption
	aes = AES encryption
	aes+tkip = support both **Note:** Do not use this value when wl0_mode=sta because supplicant mode does not seem to auto-negotiate. You must select one protocol which the access point supports (refer to the AP's specs)
wl0_wpa_psk	Password to use with WPA/WPA2 PSK (at least 8, up to 63 chars)
wl0_radius_key	Shared Secret for connection to the Radius server
wl0_radius_ipaddr	IP to connect...
wl0_radius_port	Port# to connect...
wl0_auth	**0**

4.5. Wireless Distribution System (WDS) / Repeater / Bridge

OpenWrt supports the WDS protocol, which allows a point to point link to be established between two access points. By default, WDS links are added to the br0 bridge, treating them as part of the lan/wifi segment; clients will be able to seamlessly connect through either access point using wireless or the wired lan ports as if they were directly connected.

Configuration of WDS is simple, and depends on one of two variables

NVRAM	Description
wl0_lazywds	Accept WDS connections from anyone (0:disabled 1:enabled)
wl0_wds	List of WDS peer mac addresses (xx:xx:xx:xx:xx:xx, space-separated)

For security reasons, it's recommended that you leave wl0_lazywds off and use wl0_wds to control WDS access to your AP. wl0_wds functions as an access list of peers to accept connections from and peers to try to connect to; the peers will either need the mac address of your AP in their wl0_wds list, or wl0_lazywds enabled.

Easy steps for a successful WDS:

First do it without wireless protection and then activate the protection. If you activate both you will double the pain to find a problem.

1. Configure the IPs of each AP - don't use the same! For easier maintenance you can use the same subnet.
2. Add the **other** APs MAC address to the list of allowed peers to each AP. With OpenWRT it's the variable wl0_wds. Shell to each router and do ifconfig. The MAC id for eth1 is the correct MAC id to use.
3. Disable all the unneeded services like DHCP, port forwarding, firewalling etc. **except** on the AP the has the internet connection. Remember: The other APs only act as the extended arm of the internet connected AP.

4. Configure the WLAN parameters on all APs identical. That is SSID, channel, etc. - keep it simple. If you want to try boosters etc. do this later. (In my experience the SSIDs need not be identical for WDS to work, but YMMV.)
5. Have you committed your values? Do it. And reboot.
6. Now connect a lan cable to each AP and try to ping the internet AP. It should answer. Else start checking the settings.
7. You are done. Now activate security on the devices. Optionally hide the SSID (wl0_closed=1). If WPA-PSK doesn't work chances are that a peer partner doesn't support it. Try WEP.

⚠ I experienced 20% packet loss using lazywds. It went away when disabling lazywds. You have been warned!

⚠ **NOTE:** WDS requires a br0 interface. If you broke up your bridge as detailed in "To separate the LAN from the WIFI" above, this will not just work, since you no longer have a br0. You do not need to add any interfaces to br0, the WDS interfaces will be automatically added.

4.6. WDS Routed Networks (P2P)

You might want to use routing over the WDS links, rather than bridging. You will want to break up the bridge, as explained above, and prevent wds devices from being added to the bridge by editing /etc/hotplug.d/net/01-wds.

You can then add WDS interfaces, e.g:

```
nvram set wl0_wds="00:14:12:25:CB:22 00:14:12:16:3B:28"
```

This will give you several wds0.x interfaces (note the interface names get truncated when displayed in ifconfig -- they start at wds0.49153 and increment by 0.00001). Create a set of nvram variables for ifup, e.g:

```
nvram set wds1_proto=static
nvram set wds1_ifname=wds0.49153
nvram set wds1_ipaddr=192.168.254.97
nvram set wds1_netmask=255.255.255.252
nvram set wds2_proto=static
nvram set wds2_ifname=wds0.49154
nvram set wds2_ipaddr=192.168.254.100
nvram set wds2_netmask=255.255.255.252
```

Then modify /etc/init.d/S40network to bring up these interfaces:

```
     ifup wds1
     ifup wds2
```

4.7. A note on encryption with WDS

WDS is exceptionally easy to set up. You can do it in from the web interface under Wireless. WDS will work OOB with either no encryption or WEP; other than setting your WEP key (as normal) no configuration is required.

When using WPA with WDS, the simplest method is to ensure that both routers are using the same ESSID and WDS settings; if so, you don't need to set any additional variables besides **wl0_wds**. However, some people may want to use different encryption for the WDS link than for clients, or different ESSIDs for different routers; if so, there are a number of wds_specific nvram variables that can be set; ensure that all WDS peers have the same values for these variables. If the variables are unset (as they are by default), WDS will use the same encryption settings as used for clients.

NVRAM variable	**Description**
wl0_wds_wpa_psk	Your wireless password
wl0_wds_akm	The key type (i.e. psk)
wl0_wds_crypto	The algorithm (i.e. aes)
wl0_wds_ssid	The ssid (has to be the same at both ends, if used - see below)

If using WDS between routers with different ESSIDs, you should all of their **wl0_wds_ssid** variables to the ESSID of *one* of the routers, so that they will be able to talk to each other.

Note that it appears that there is a bug in nas that prevents WPA2 from working properly with WDS. It is known that WPA1 works.

Remember that the non-free package NAS must be installed for WPA to work. It is also noted on the forum that you may be able to use WPA1 for the WDS link and WPA2 for client PCs; however, consider that the protection offered by WPA is only as good as the weakest link in the chain. Any data sent over the WDS link (including connections originating from client PCs connected to the satellite AP) will be vulnerable to an attack on WPA1.

4.8. Wireless client / wireless bridge

The only thing you have to do is to switch the WL mode like with the bridge:

```
nvram set wl0_mode=wet
```

For more information, see ClientModeHowto.

5. Basic system configuration and usage

5.1. busybox - The Swiss Army Knife of Embedded Linux

Provides replacements for most of the utilities usually found in GNU fileutils and shellutils. For details see here

5.2. cron - job scheduler

See HowtoEnableCron.

5.3. syslog - Logging

To read the syslog messages, use the **logread** command. See MiniHowtos to set up remote logging.

5.4. dropbear - Secure Shell server

For SSH login without password, put your keys in /etc/dropbear/authorized_keys. See DropbearPublicKeyAuthenticationHowto.

5.5. iptables - Firewall

The rules and some small samples for your firewall can be found in /etc/firewall.user. For RC5 and earlier if you want to make changes to this file you have to remove it first since it is actually a symlink to /rom/etc/firewall.user, see the section Editing files in OpenWrtDocs/Using.

Be sure to read the notes about the firewall rules before changing anything. The important thing to note is that if you setup port forwarding, you won't be able to see the changes inside the router's LAN. You will have to access the router from outside to verify the setup.

As of RC9 the file /etc/firewall.user reads

```
#!/bin/sh
# Copyright (C) 2006 OpenWrt.org
iptables -F input_rule
iptables -F output_rule
iptables -F forwarding_rule
iptables -t nat -F prerouting_rule
iptables -t nat -F postrouting_rule
# The following chains are for traffic directed at the IP of the
# WAN interface
iptables -F input_wan
iptables -F forwarding_wan
iptables -t nat -F prerouting_wan
### Open port to WAN
## -- This allows port 22 to be answered by (dropbear on) the router
# iptables -t nat -A prerouting_wan -p tcp --dport 22 -j ACCEPT
# iptables        -A input_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_wan -p tcp --dport 8080 -j DNAT --to 192.168.1.2:80
# iptables        -A forwarding_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_wan -j DNAT --to 192.168.1.2
# iptables        -A forwarding_wan -d 192.168.1.2 -j ACCEPT
```

The first section, **Open port to WAN** shows an example of opening a port for your router running OpenWRT to listen to and accept. In the case given, it will open up port 22 and accept connections using dropbear (the SSH server). Just delete the **#** sign in front of the two rules to enable access.

If you wanted to open up any other ports for the router to listen to, just copy those two lines and change just the port number from 22 to something else.

The second section, **Port forwarding** is for accepting incoming connections from the WAN (outside the router) and sending the requests to a networked device on your LAN (inside your router).

Before setting up any port forwarding, you'll have to install some OpenWRT packages first, such as iptables-nat and ip (any others?).

In the example provided, if someone on the Internet were to connect to your router on port 8080, it would forward them to port 80 on whatever computer / device had the IP address of 192.168.1.2.

If you are running a webserver on that address, and want to listen on port 80 instead, change the 8080 on the first line.

The same is true for any other ports you'd want to forward to your LAN. Just follow the example as a guide.

If you would like to have the router loopback forwarded ports, you need to add the following code to /etc/firewall.user. Loopback allows a computer on your LAN to hit your external IP address and have the packet forwarded back as if it had come from the outside. The default OpenWrt (iptables) installation does not allow this.

```
iptables -t nat -A prerouting_rule -d 100.100.100.100 -p tcp --dport 80 -j DNAT --to 192.168.0.2
iptables -A forwarding_rule -p tcp --dport 80 -d 192.168.0.2 -j ACCEPT
iptables -t nat -A postrouting_rule -s 192.168.0.0/24 -p tcp --dport 80 -d 192.168.0.2 -j
MASQUERADE
```

There is an explanation for what these lines mean on this form about half way down. The example above loopbacks all traffic on port 80 directed from the LAN to the external IP address 100.100.100.100 back to 192.168.0.2. You need to copy these three lines and change the port number for every port needing loopback. You would usually use this with an existing port forwarding rule described. For example:

```
iptables -A forwarding_wan -p tcp --dport 80 -d 192.168.0.2 -j ACCEPT
```

If you are using x-wrt to setup port forwarding this rule will be created in /etc/config/firewall and will look like the following:

```
forward:proto=tcp dport=80:192.168.0.2
```

These instructions only work for single port numbers. If anyone knows how to loopback a port range please post the instructions here.

The last section, **DMZ** is sending all connections to a port not specified in the rules above to a certain IP address. If you do decide to use this, it would be a good idea to have a firewall managing the ports on the destination. The DMZ can be considered a simple way to let another computer handle the firewall rules, if you don't want to configure them on OpenWRT and at the same time you want to send all connections to one device.

Once you're finished making changes to your firewall, restart it by running the init script:

```
/etc/init.d/S45firewall restart
```

Remember to test the changes outside your LAN!

Finally, if you wish to dig deeper into how iptables work under the rule/chain structure of OpenWRT, see OpenWrtDocs/IPTables

5.6. dnsmasq - DNS and DHCP server

Dnsmasq is a lightweight, easy to configure DNS forwarder and DHCP server.

Documentation can be found at OpenWrtDocs/dnsmasq.

5.7. Time

Most devices supported by OpenWrt have no real-time clock hardware onboard, and must get the date and time at boot or use the default of 2000-01-01.

You must have the correct time to use OpenVPN on OpenWrt. The same applies to other tools using CA certificates such as wget and curl.

You may use either *ntpclient*, *rdate*, *htpdate* or *openntpd*. Only *rdate* is included by default.

rdate

The *rdate* command synchronises the system time to the time on a remote host using the time protocol on TCP port 37 (the time protocol has been superseded by the Network time protocol (NTP)). It is normally used once during boot, and then the kernel maintains the time based on the processor oscillator. It will slowly drift. *rdate* is part of the *busybox* package and is already installed.

Create the file `/etc/init.d/S55rdate` with the contents:

```
#!/bin/sh
/usr/sbin/rdate -s HOST
```

replacing HOST with the IP address or host name of the time server, E.G.

```
#!/bin/sh
/usr/sbin/rdate -s timeserver.example.net
```

Then make the file executable:

```
chmod a+x /etc/init.d/S55rdate
```

then either reboot or run it once:

```
/etc/init.d/S55rdate
```

Make sure any software that is loaded in the boot sequence and which requires the correct time is started later than S55rdate. Remember that DNS host names will not be resolved before S50dnsmasq has been run, so be careful if changing S55rdate to run earlier in the boot sequence.

If your router is not rebooted very regularly you may wish to add updating the time to the crontab. The following will update the time each day at 06.30 AM.

Edit the crontab file by typing:

```
crontab -e
```

Then add this line to the file:

```
30 6 * * * /usr/sbin/rdate -s HOST
```

Again replacing HOST with the IP address or host name of the time server.

ntpclient

ntpclient will synchronize the system time using the NTP protocol when the internet connection is established. To set it up follow this instructions :

Set the *ntp_server* NVRAM variable to your preferred NTP server (for example the NTP server of your ISP; if no server is set, *ntpclient* will use *pool.ntp.org* as default):

```
nvram set ntp_server=ntp.my-isp.net
nvram commit
```

Install the *ntpclient* package in the web interface or using the command

```
ipkg install ntpclient
```

ntpclient will now update the system time each time the WAN connection is established. To set the time manually use this command line

```
/usr/sbin/ntpclient -c 1 -d -s -h ntp.my-isp.net
```

or reboot the router. (the *-d* option just prints some info about what is going on, you can leave it out)

5.8. Timezone

Without a time zone set, OpenWrt will display UTC.

To set a time zone use the `/etc/TZ` file. Copy & paste the time zones from the table below into the file. In this example it's done with the `echo` command.

```
echo "CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00" > /etc/TZ
```

NOTE: This sets the time zone for CET/CEST (Central European Time UTC+1 / Central European Summer Time UTC+2) and the starting (5th week of March at 02:00) and endtime (5th week of October at 03:00) of DST (Daylight Saving Time).

More can be found here http://leaf.sourceforge.net/doc/guide/buci-tz.html#id2594640 and http://openwrt.org/forum/viewtopic.php?id=131.

Note: When using openNTPd on RC6, with or without X-wrt, it seems the above mentioned method doesn't survive reboot. I actually use vi created a /etc/TZ file with relevant timezone and it works well.

Better use this:

```
nvram set time_zone="CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00"
nvram commit
```

Given that create a script /etc/init.d/tz with the following content to remember the timezone after reboot. Note that this script uses the CET/CEST if no nvram value is present, adjust it to your needs. Do not forget to chmod a+x the tz-file.

```
#!/bin/sh

tz=$(nvram get time_zone)
tz=${tz:-"CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00"}
echo $tz > /etc/TZ
```

Examples of timezone values:

Australia	Melbourne,Canberra,Sydney	AEST-10AEDT-11,M10.5.0/02:00:00,M3.5.0/03:00:00
	Perth	AWST-8AWDT-9,M12.1.0,M3.5.0/03:00:00
	Brisbane	AEST-10
	Adelaide	ACST-9:30ACDT-10:30,M10.5.0/02:00:00,M3.5.0/03:00:00
	Darwin	ACST-9:30
	Hobart	AEST-10AEDT-11,M10.1.0/02:00:00,M3.5.0/03:00:00
Europe	Amsterdam, Netherlands	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Athens, Greece	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
	Barcelona, Spain	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Berlin, Germany	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Brussels, Belgium	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Budapest, Hungary	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Copenhagen, Denmark	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Dublin, Ireland	GMT+0IST-1,M3.5.0/01:00:00,M10.5.0/02:00:00
	Geneva, Switzerland	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Helsinki, Finland	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
	Kyiv, Ukraine	EET-2EEST,M3.5.0/3,M10.5.0/4
	Lisbon, Portugal	WET-0WEST-1,M3.5.0/01:00:00,M10.5.0/02:00:00

	London, Great Britain	GMT+0BST-1,M3.5.0/01:00:00,M10.5.0/02:00:00
	Madrid, Spain	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Oslo, Norway	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Paris, France	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Prague, Czech Republic	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Roma, Italy	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Moscow, Russia	MSK-3MSD,M3.5.0/2,M10.5.0/3
	Sofia, Bulgaria	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
	St.Petersburg, Russia	MST-3MDT,M3.5.0/2,M10.5.0/3
	Stockholm, Sweden	CET-1CEST-2,M3.5.0/02:00:00,M10.5.0/03:00:00
	Tallinn, Estonia	EET-2EEST-3,M3.5.0/03:00:00,M10.5.0/04:00:00
Warsaw, Poland	CET-1CEST,M3.5.0,M10.5.0/3	
New Zealand[2]	Auckland, Wellington	NZST-12NZDT-13,M10.1.0/02:00:00,M3.3.0/03:00:00
USA & Canada[1]	Hawaii Time	HAW10
	Alaska Time	AKST9AKDT,M3.2.0,M11.1.0
	Pacific Time	PST8PDT,M3.2.0,M11.1.0
	Mountain Time	MST7MDT,M3.2.0,M11.1.0
	Mountain Time (Arizona, no DST)	MST7
	Central Time	CST6CDT,M3.2.0,M11.1.0
	Eastern Time	EST5EDT,M3.2.0,M11.1.0
	Atlantic Time	AST4ADT
	Atlantic Time (New Brunswick)	AST4ADT,M4.1.0/00:01:00,M10.5.0/00:01:00
	Newfoundland Time (Updated DST for 2007)	NST+3:30NDT+2:30,M3.2.0/00:01:00,M11.1.0/00:01:00
Asia	Jakarta	WIB-7
	Singapore	SGT-8
	Hong Kong	HKT-8
Ulaanbaatar, Mongolia	ULAT-8ULAST,M3.5.0/2,M9.5.0/2	
Central and South America	Brazil, São Paulo	BRST+3BRDT+2,M10.3.0,M2.3.0
	Argentina	UTC+3
	Central America	CST+6

Please update and include your time zone. You can find more on time zones on timeanddate.com.

[1] in August of 2005, the United States President Bush passed the Energy Policy Act, which, among other things, changes the time change dates for daylight saving time from the first Sunday in April to the second Sunday in March and from the last Sunday in October to the first Sunday in November. This pattern starts in 2007, however, and Congress still has time to revert the DST back. As such, these changes have not yet been incorporated into mainline uClibc (which provides the time functions for the C library used by OpenWrt). Therefore, it might be a good idea to change `/etc/TZ` explicitly (around mid-November 2006) to reflect this change (i.e., instead of `EST5EDT` write `EST5EDT,M3.2.0,M11.1.0`).

[2] on 30.April 2007, Daylight Savings Time was changed for New Zealand as well. Hence from September 2007 on, clocks will go forward an hour a week earlier than usual - on the last Sunday in September - and back an hour on the first Sunday in April, instead of the third Sunday in March. So use `NZST-12NZDT-13,M9.5.0/02:00:00,M4.1.0/03:00:00` instead of the string above. Ref: http://www.dia.govt.nz/diawebsite.nsf/wpg_URL/Services-Daylight-Saving-Daylight-saving-to-be-extended?OpenDocument

Here is the command to type for each time zone in the continential US:

```
echo "EST5EDT,M3.2.0,M11.1.0" >/etc/TZ
echo "CST6CDT,M3.2.0,M11.1.0" >/etc/TZ
echo "MST7MDT,M3.2.0,M11.1.0" >/etc/TZ
echo "PST8PDT,M3.2.0,M11.1.0" >/etc/TZ
echo "AKST9AKDT,M3.2.0,M11.1.0" >/etc/TZ
```

As explained above, you could also set this in the NVRAM:

```
rm -f /etc/TZ
nvram set time_zone="EST5EDT,M3.2.0,M11.1.0"
nvram set time_zone="CST6CDT,M3.2.0,M11.1.0"
nvram set time_zone="MST7MDT,M3.2.0,M11.1.0"
nvram set time_zone="PST8PDT,M3.2.0,M11.1.0"
nvram set time_zone="AKST9AKDT,M3.2.0,M11.1.0"
nvram commit
```

See http://astronomy.physics.tamu.edu/Java/Tools/Misc/Clock/zones.html for other time zones.

6. HOWTOs / Additional Configuration

See also:

- OpenWrtHowTo
- OpenWRT Faq.
- OpenWrtDocs/IPTables for a more detailed explanation of iptables under OpenWRT
- tcpdumpHowTo How to set up tcpdump as a daemon

OpenWrtDocs/Configuration (ostatnio edytowane 2007-08-02 21:35:04 przez sorenstoutner)