#### You are here: DD-WRT wiki mainpage / Scripting / Useful Scripts

Certain scripts can enhance the function of your router with DD-WRT, giving it more features or customizing it towards your needs.

To find out how to load scripts onto the router, see Startup Scripts.

# Contents

- <u>1 Modifying \$PATH at Startup</u>
- <u>2 LED Scripts</u>
- <u>3 Miscellaneous Scripts</u>
  - ♦ <u>3.1 Backup settings and restore them</u>
  - ◆ <u>3.2 Clear ttraff (WAN bandwidth graph) nvram</u> <u>data</u>
    - ◊ <u>3.2.1 The other way (NEW, 2012)</u>
  - ◆ <u>3.3 Compress the Firewall Script (to reduce</u> <u>nvram usage)</u>
  - ◆ <u>3.4 Web Server Wake-up</u>
  - ◆ <u>3.5 Auto Random MAC Address</u>
  - ♦ <u>3.6 Wireless Network Scanner (awk -f scanner)</u>
  - ♦ 3.7 Wireless Network Scanner (working on <u>DD-WRT v24)</u>
  - ◆ <u>3.8 Name-based WOL (wake.sh)</u>
  - ◆ <u>3.9 Name-based WOL with dnsmasq</u>
  - ◆ <u>3.10 Automatic Connection Repair</u>
  - ◆ <u>3.11 Modifying \$PATH Manually (path.sh)</u>
  - ◆ <u>3.12 View Logfile in Browser without Local</u>
    - Syslogd (log.sh)
      - ◊ <u>3.12.1 First Method: Script Generated</u> <u>Live-content</u>
      - ◊ <u>3.12.2 Second Method: Static Generated</u> <u>HTML</u>
  - ◆ <u>3.13 Speak Your Signal Strength</u>
  - ◆ <u>3.14 Small Security Script (Firewall)</u>
  - ◆ <u>3.15 Secure remote management for a WAP</u> (Firewall scripts)
  - ♦ <u>3.16 Directory Listing for DD-WRT Micro</u>
  - ◆ <u>3.17 Reset Wireless Radio</u>
  - ♦ <u>3.18 Display Connection Counts Per IP</u>
    - ◊ <u>3.18.1 For local IP's</u>
    - ◊ <u>3.18.2 For all IP's</u>
  - ♦ <u>3.19 L2TP Fix for HOT users (obsolete)</u>
  - ◆ <u>3.20 Email Bandwidth Usage Daily</u>
  - ♦ <u>3.21 Update Allowed Remote IP Range by DNS</u> <u>lookup</u>
  - ♦ 3.22 DynDNS Updates Using Curl (with <u>HTTPS/SSL Support</u>)
  - ◆ <u>3.23 Other Useful Misc. Scripts</u>
- <u>4 See Also</u>

### Modifying \$PATH at Startup

This will add whatever paths you want for \$PATH and \$LD\_LIBRARY\_PATH before the default system path. Change the paths to whatever you like. Have a good reason for doing this, it should be considered a hack until the feature is implemented permanently.

```
rm -f /tmp/newProfile
head -n1 /etc/profile | sed s!=!=/mmc/bin:/whatever/bin:! >> /tmp/newProfile
tail -n1 /etc/profile | sed s!=!=/mmc/lib:/whatever/lib:! >> /tmp/newProfile
mount --bind /tmp/newProfile /etc/profile
```

If you're adding /mmc/lib before the system library, in some circumstances you'll also need to do this on startup (after ensuring that the ldconfig on /mmc is up to date and happy):

```
mount --bind /mmc/etc/ld.so.conf /etc/ld.so.conf
mount --bind /mmc/etc/ld.so.cache /etc/ld.so.cache
```

Note: Only do this if you're receiving segmentation faults or your applications are failing to run, and even then only if you feel that this hack is imperative. Also note that if you're attempting this with <u>Optware</u>, the files are ld-opt.so.conf and ld-opt.so.cache

Another alternative to the above mentioned way of altering things is to simply copy /etc/profile to /jffs/etc, change PATH and LD\_LIBRARY\_PATH and bind the profile to the original location.

```
mkdir /jffs/etc
cp /etc/profile /jffs/etc/profile
cd /jffs/etc
vi profile (change everythings that suite your needs and save it with :x)
```

Afterward put the line below inside a startup script

mount --bind /jffs/etc/profile /etc/profile

Make sure you're familiar with what you're doing before attempting this, if you end up seeing a lot of segmentation faults when running things like ls, cat, cp, etc, than you'll want to either adjust the above commands, or else put those things into a script and run them manually when you enter your shell.

# **LED Scripts**

See LED Scripts

# **Miscellaneous Scripts**

## Backup settings and restore them

Credit to hardwarewizard and frater

#### Reference Thread

```
#!/bin/sh
*****
#
  Script Requirements
#
#
 Files:
#
    vars_to_skip
#
     vars_preferred
#
#
  Programs:
     curl
#
*****
*****
# setup variables
#
# DATE
           - Date
          – Mac address
# MAC
# FILE
            - File Name Prefix
           - Current Directory
# CUR_DIR
# transfer
            - FTP Transfer ON/OFF (Default is OFF)
# FOLDER
         - Location where backup scripts are stored
# VARFILE - Location & Name of Temp File
# TO_ALL - Location & Name of script File with all nvram variables
# TO_INCLUDE - Location & Name of script File with essential nvram variables
# TO_EXCLUDE - Location & Name of script File with dangerous nvram variables
# TO_PREFERRED - Location & Name of script File with preferred nvram variables
****
DATE=`date +%m%d%Y`
MAC=`nvram get lan hwaddr | tr -d ":"`
FILE=${MAC}.${DATE}
CUR DIR=`dirname $0`
transfer=0
FOLDER=/opt/vars/backups
VARFILE=/opt/tmp/all_vars
TO_ALL=${FOLDER}/${MAC}.${DATE}.all.sh
TO_INCLUDE=${FOLDER}/${MAC}.${DATE}.essential.sh
TO_EXCLUDE=${FOLDER}/${MAC}.${DATE}.dangerous.sh
TO_PREFERRED=${FOLDER}/${MAC}.${DATE}.preferred.sh
****
#FTP Login information change to your info
****
FTPS=ftp://192.168.1.100/backups
USERPASS=user:pass
*****
# read command line switches
#
#
   example command lines
#
#
   ./backupvars.sh -t
#
#
  The above command with use the user and password and
  server information embedded in this script.
#
   (See FTP Login information above)
#
#
#
```

```
#
   ./backupvars.sh -t -u user:pass -f ftp://192.168.1.100/backups
#
   The above command with use the user and password and
#
#
   server information from the command line
#
****
while getopts tu:f: name
do
 case $name in
 t) transfer=1;;
    USERPASS="$OPTARG";;
 11)
 f) FTPS="$OPTARG";;
     printf "Usage: %s: [-t] [-u username:password] [-f ftpserver]\n"
 ?)
     exit 2;;
 esac
done
shift $(($OPTIND - 1))
****
#create NVRAM variale list and write to /opt/tmp/all_vars
****
nvram show 2>/dev/null | egrep '^[A-Za-z][A-Za-z0-9_\.\-]*=' | awk -F = '{print $1}' | sort -r -u
*****
# Write header to restore scripts
****
echo -e "#!/bin/sh\n#\necho \"Write variables\"\n" | tee -i ${TO_EXCLUDE} | tee -i ${TO_PREFERRED
****
# scan NVRAM variable list and send variable to proper
# restore script
****
cat ${VARFILE} | while read var
do
 pref=0
### replaced with next line by Andon Man?ev : if echo "${var}" | grep -q -f "${CUR_DIR}/vars_tc
if cat "${CUR_DIR}/vars_to_skip" | grep -q "${var}" ; then
   bfile=$TO_EXCLUDE
 else
  bfile=$TO_INCLUDE
   pref=`echo "${var}" | grep -cf "${CUR_DIR}/vars_preferred"`
 fi
 # get the data out of the variable
 data=`nvram get ${var}`
 # write the var to the file and use \ for special chars: (\$`")
 echo -en "nvram set ${var}=\"" | tee -ia ${TO_ALL} >> ${bfile}
 echo -n "${data}" | sed -e 's/[$`"\]/\\&/q' | tee -ia ${TO_ALL} >> ${bfile}
 echo -e "\"" | tee -ia ${TO_ALL} >> ${bfile}
 if [ ! ${pref} == 0 ]; then
   echo -en "nvram set ${var}=\"" >> ${TO_PREFERRED}
   echo -n "\{data\}" | sed -e 's/[\hat{ }"\]/\\(q' >> ${TO_PREFERRED}}
   echo -e "\"" >> ${TO_PREFERRED}
 fi
done
****
# cleanup remove /opt/tmp/all_vars
```

```
# uncomment to remove file
*****
# rm ${VARFILE}
*****
# Write footer to restore script
*****
echo -e "\n# Commit variables\necho \"Save variables to nvram\"\nnvram commit" | tee -ia ${TO_A
****
\ensuremath{\texttt{\#}} Change permissions on restore scripts to make them
# executable
****
chmod +x ${TO_INCLUDE}
chmod +x ${TO_PREFERRED}
chmod +x ${TO_EXCLUDE}
chmod +x ${TO_ALL}
****
# Compress restore scripts and send them to ftp server
****
if [ ${transfer} -ne 0 ] ; then
 tar cpf - -C / "${TO_INCLUDE}" 2>/dev/null | gzip -c | /opt/bin/curl -s -u ${USERPASS} "${FTPS
 tar cpf - -C / "${TO_PREFERRED}" 2>/dev/null | gzip -c | /opt/bin/curl -s -u ${USERPASS} "${FT
 tar cpf - -C / "${TO_EXCLUDE}" 2>/dev/null | gzip -c | /opt/bin/curl -s -u ${USERPASS} "${FTPS
 tar cpf - -C / "${TO_ALL}" 2>/dev/null | gzip -c | /opt/bin/curl -s -u ${USERPASS} "${FTPS}/${
fi
```

#### The vars\_to\_skip file

DD BOARD ^board browser\_method ^cfe ct modules custom\_shutdown\_command ^def\_ ^default dist\_type dl\_ram\_addr early\_startup\_command ^et0 ^et1 ^ezc generate\_key gozila\_action gpio ^hardware ^is\_ ^kernel\_ lan\_default ^lan\_hw ^lan\_ifname landevs manual\_boot\_nv misc\_io\_mode need\_commit ^os\_

Backup settings and restore them

overclocking pa0maxpwr phyid\_num pmon\_ver pppd\_pppifname pppoe\_ifname pppoe\_wan\_ifname primary\_ifname probe\_blacklist regulation\_domain rescue reset scratch sdram ^sh\_ ^skip sshd\_dss\_host\_key sshd\_rsa\_host\_key startup\_command ^wan\_default ^wan\_hw ^wan\_if ^wan\_vport ^wandevs web\_hook\_libraries ^wifi\_ w10.1\_hwaddr w10.2\_hwaddr w10.3\_hwaddr wl0\_hwaddr wl0 ifname wl0 radioids ^wl ^wlan\_

The vars\_preferred file. This is the file you add your vars to that you want to backup.

daylight\_time
time\_zone

this is an updated version

# Clear ttraff (WAN bandwidth graph) nvram data

The ttraff daemon can fill up a couple hundred bytes of nvram space every month. This may not seem like much but nvram is only ~32KB total and is full of lots of other data. Disabling ttraff and clearing it's old nvram data is sometimes needed for devices with complex configurations, or to keep the router stable. This script will clear **all** of ttraff's traffic data from nvram whereas using the ttraff GUI button to delete it still leaves the current month's variable.

for i in `nvram show | grep traff- | cut -f1 -d=""`; do nvram unset \$i; done

#### The other way (NEW, 2012)

If you want to keep always one year of history, use the one below in your startup script to prune old data (no cron needed, and it doesn't require a correct date is set e.g. with ntpd: the script unsets nvram entries elder than 12 monthes before the most recent data available, i.e. 13 data sets, current month plus last 12). As the aim is to reclaim nvram storage and as the startup script is stored in nvram, I tried to keep it the slimmest possible (no indentation, no function whose declaration would have store more space than duplicated commands, single command per line to spare one out of the two "; " bytes separator (single line return instead), no [tmp]file stored and single letter variables T for Traff or Threshold, D for Dates and N for nvram. Any other shrink trick is welcome. Should you run your dd-wrt unattended for years with traffmon enabled, you should use this script or you'll run one day or the other into a nvram outage and its fancy effets as wireless changing ciphering mode etc... You can adjust how long the log you'll keep will be by changing the integer "100" in the eleventh line: the first loop searches the max (say current) data set date, in YYYMM format, then the T=`expr ...` line sets the lower (threshold) that will be kept, e.g 200 for 2 years, 109 for 1 year plus 9 monthes. A last word, about the current data set date "detection". Initial T and subsequent loop could have been replaced by a simple T=`date +%Y%m` or more condensed the script could have begun with T=`expr`date +%Y%m` - 100` (26 chars against 31 -don't forget threshold setting-, but not checked), but as year start on 1970 on startup until a true ntp date is available, the script would not work reliabily - although this could be used with the script as custom script ran from cron, say monthly, with the help of a replacement check that year date is not 1970 (or better: higher than eldest data set, to protect against any firmware change, but here the script size increases).

```
#!/bin/sh
T=0
D=`nvram show|grep traff-|cut -f1 -d=|awk -F"-" '{print $3,$2}'|cut -b-4,6-`
for d in $D
do
if [ $T -lt $d ]
then
T=$d
fi
done
T=`expr $T - 100`
for d in $D
do
if [ $T -gt $d ]
then
N=traff-`echo ${d}|cut -b5-`-`echo ${d}|cut -b-4`
nvram unset $N
fi
done
```

Thank you JLL and Glenfarclass and Cardhu for the help --Bib 02:13, 12 November 2012 (CET)

### Compress the Firewall Script (to reduce nvram usage)

If you have a large firewall script you can use this script to compress it with gzip to use less nvram space. See <u>this thread</u> for full usage info.

```
# Compress Firewall
nvram set pH_fw="`nvram get rc_firewall | gzip | uuencode -m /dev/stdout`"
nvram set rc_firewall="nvram get pH_fw | uudecode -o /tmp/pH_fw.gz;gunzip /tmp/pH_fw.gz;chmod +x
/tmp/pH_fw;/tmp/pH_fw"
```

```
nvram show >/dev/null
# Decompress Firewall
nvram get pH_fw | uudecode -o /tmp/pH_fw.gz
nvram unset pH_fw
gunzip /tmp/pH_fw.gz
nvram set rc_firewall="`cat /tmp/pH_fw`"
nvram show >/dev/null
# Editing the Firewall
vi /tmp/pH_fw
# finish editing with vi before running the rest
nvram set rc_firewall="`cat /tmp/pH_fw`"
nvram show >/dev/null
```

### Web Server Wake-up

• Wakes up your web server when the router receives a request from the internet. Credits from <u>here.</u> RET value from that script did not work here, using if ping else statement below. --rasos 29NOV2010

Please note: syslogd needs to be on, logging enabled, with log level set high, and "accepted" on. Following the example script, replace target and MAC values with those of your LAN web server's network information and for "\$WOL -i xxx.xxx.255", replace xxx.xxx.255 with your LAN network broadcast address.

```
#!/bin/sh
```

```
INTERVAL=5
NUMP=3
OLD=""
WOL=/usr/sbin/wol
TARGET=192.168.3.18
MAC=00:26:18:CF:E1:E1
LOGFILE="/tmp/www/wol.log"
while sleep $INTERVAL; do
 NEW=`awk '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT=80/ {print }' /var/log/messages | tail -1`
  SRC=`awk -F'[=| ]' '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT=80/ {print }' /var/log/messages | tai
 LINE=`awk '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT=80/' /var/log/messages`
  if [ "$NEW" != "" -a "$NEW" != "$OLD" ]; then
    if ping -qc $NUMP $TARGET >/dev/null; then
      echo "$TARGET was accessed by $SRC and is alive at" `date` >> $LOGFILE
    else
      echo "$SRC causes wake on lan at" `date` >> $LOGFILE
      $WOL -i 192.168.3.255 -p 7 $MAC >> $LOGFILE
     sleep 5
    fi
   OLD=$NEW
 fi
done
```

Unfortunately that script wasn't working with V24-SP2's logging. I rewrote it to use dmesg and added some logging messages to help. Thanks go to rasos for the initial script!

#!/bin/sh
#Enable JFFS2 and place script in /jffs/ then run on startup in web interface.
#You can check the log from http://192.168.1.1/user/wol.html

Compress the Firewall Script (to reduce nvram usage)

```
INTERVAL=5
NUMP=3
OLD=""
PORT=80
WOLPORT=9
TARGET=192.168.1.101
BROADCAST=192.168.1.255
MAC=00:11:22:33:44:55
WOL=/usr/sbin/wol
LOGFILE="/tmp/www/wol.html"
echo "<meta http-equiv=\"refresh\" content=\"10\">" > $LOGFILE
echo "AUTO WOL Script started at" `date` "<br>> $LOGFILE
while sleep $INTERVAL; do
NEW=`dmesg | awk '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT='"$PORT"'/ {print }' | tail -1`
SRC=`dmesg | awk -F'[=| ]' '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT='"$PORT"'/ {print $7}' | tail -
LINE=`dmesg | awk '/ACCEPT/ && /DST='"$TARGET"'/ && /DPT='"$PORT"'/'`
if [ "$NEW" != "" -a "$NEW" != "$OLD" ]; then
   if ping -qc $NUMP $TARGET >/dev/null; then
      echo "NOWAKE $TARGET was accessed by $SRC and is already alive at" `date` "<br>">> $LOGFILE
   else
      echo "WAKE $SRC causes wake on lan at" `date` "<br>>> $LOGFILE
      $WOL -i $BROADCAST -p $WOLPORT $MAC >> $LOGFILE
      echo "<br>> $LOGFILE
      sleep 5
   fi
   OLD=$NEW
fi
done
```

### Auto Random MAC Address

• This script will change your eth1 MAC address to a random address, then it will apply it to the system and restart the interfaces.

```
#!/bin/ash
MAC=`(date; cat /proc/interrupts) | md5sum | sed -r 's/^(.{10}).*$/\1/; s/([0-9a-f]{2})/\1:/g; s/
echo "00:${MAC}"
ifconfig eth1 hw ether 00:${MAC}
nvram set def_hwaddr="00:${MAC}"
nvram set wan_hwaddr="00:${MAC}"
stopservice wan
startservice wan
```

You may wish to also download *curl* (see ipkg), and use it to restart your modem, as some MAC changes may not reflect until your modem "sees" a new address, and they typically only do this when starting up.

Note: curl is sometimes problematic to install. You should use ipkg -force-depends

An example, to restart a Motorola Surfboard SB4100 cable model is:

curl -s -d "BUTTON\_INPUT=Restart+Cable+Modem" http://192.168.100.1/configdata.html

To restart a Motorola SB5101:

curl -d ResetReq=1 http://192.168.100.1/goform/RgConfig

I added the following lines to the end of the above to restart a Motorola SB5120 (no curl required!!) and reboot. Running this script in cron, and my ISP won't automatically recover without the following:

nvram commit & sleep 5 && wget http://192.168.100.1/reset.htm?reset\_modem=Restart+Cable+Modem
reboot

Don't know the reason but Motorola SB5101 only re-started with the two following lines:

```
curl -v -d "RestoreFactoryDefault=1&ResetReq=1" http://192.168.100.1/goform/RgConfig curl -d ResetReq=1 http://192.168.100.1/goform/RgConfig
```

### Wireless Network Scanner (awk -f scanner)

```
cat - > scanner
# Show scanresults in consistent order with graphical bars.
# To be run via telnet to WRT54g running modified firmware.
# Do the following. Use your own router address instead of 192.168.1.1 on the following lines
# Login via telnet:
    telnet 192.168.1.1
#
# a simple test to make sure you can run this script, type:
   wl scan; wl scanresults
#
# and make sure you can run those commands. If not this program will not work.
# If you succeeded with the scanresults then
# copy and paste this entire text into the terminal window
# (the cat - > scanner line will copy the rest of the file into a file named 'scanner')
# and then hit return and then ctrl-c to close the file.
# then just run script by typing the following line:
#
    awk -f scanner
# I hereby release this into the public domain. Justin Jones, 2005
# Jan. '07 corrected bug from '06 improvement.
BEGIN{
IGNORECASE = 1;
command = "wl scan 2> /dev/null ; wl scanresults 2> /dev/null";
red = "\x1b[31m"; green = "\x1b[32m";
                             yellow = "\x1b[33m";
greenback="\x1b[42m";
                              blue = "\x1b[34m";
cyan = " \ x1b[36m";
                            white = "\xlb[37m";
reset = "\xlb[0m";
clear = "\xlb[2J";
blueback = "\x1b[44m";
whiteback = "\x1b[47m";
underscore = "\x1b[4m";
home = "\x1b[0;0H";
                               erase2end = "\x1b[K";
cName = white;
                                cSignal = green;
cNoise = red;
                                cCaps = green;
cStrengthLow = blue blueback; cChannel = green;
cStrengthMed = white whiteback;
cStrengthHi = green greenback;
```

```
print clear;
```

#### Auto Random MAC Address

cStrengthAged = red;

```
for(;;)
{
 while (command|getline)
 if(/^SSID/) { name = ; rssi = ;noise= ; rssi=""; noise="";channel="";bssid="";caps=""}
 if(/^Mode/) {rssi = ;noise= ; channel = }
 if(/^BSSID/) {bssid = ; caps = " "" "" "" "" " }
 if(/^Supported/)
       {
       name[bssid] = name
       rssi[bssid] = rssi
       noise[bssid]= noise
       channel[bssid] = channel
       caps[bssid] = caps
       }
 }
 close (command)
 printf home;
 ln = 0;
 print white " Name BSSID Signal Noise Channel Type";
 for (x in name)
 {
       #arbitrary strength calc through trial and error... modify as you wish:
       sigstrength = ((rssi[x] - noise[x])*1.5) + ((rssi[x] +90)*1.5);
       if (sigstrength <1) sigstrength=0;
       cStrength = cStrengthLow;
       if(sigstrength>4) cStrength = cStrengthMed;
       if(sigstrength>7) cStrength = cStrengthHi;
       if(age[x]=0) cStrength = cStrengthAged;
       fmt = "%s%-15s %s%0"sigstrength"d "reset erase2end "\n %s %s%-4d %s%-4d %s%2s
       printf fmt, cName,name[x],cStrength,0,x,cSignal,rssi[x],cNoise,noise[x],cChannel, channel
       rssi[x] = "-1000 xxxx";
       ln++;
 }
 if (ln ==0) print red "No Results - Do you have wl scan capability? \nThis program depends on '
 print erase2end;
}
}
```

# Wireless Network Scanner (working on DD-WRT v24)

I took the above script and tweaked it to work in DD-WRT v24 firmware, with the "wl" command.

To run just copy and paste in a console (telnet or ssh) or save as a "scanner.sh" and run as ./scanner.

```
#!/bin/sh
awk -F"[][]" '
BEGIN{
    IGNORECASE = 1;
    command = "site_survey 2>&1";
    red = "\x1b[31m"; green = "\x1b[32m";
    greenback="\x1b[42m"; yellow = "\x1b[33m";
```

```
cyan = "\x1b[36m";
                               blue = "\x1b[34m";
blueback = "\x1b[44m";
whiteback = "\x1b[47m";
underscore = "\x1b[4m";
clear = "\x1b[2J";
home = " \times 1b[0; 0H";
                               erase2end = "\x1b[K";
cName = white;
                                cSignal = green;
cNoise = red;
                                cCaps = green;
cStrengthLow = blue blueback; cChannel = green;
cStrengthMed = white whiteback;
cStrengthHi = green greenback;
cStrengthAged = red;
print clear;
for(;;)
 {
 while (command|getline)
    if ( == "") continue;
   bssid=;
       name[bssid] = ;
       rssi[bssid] = ;
       noise[bssid]= ;
       channel[bssid] = ;
       caps[bssid] = ;
       age[bssid] = 1;
  }
 close(command);
 printf home;
 ln = 0;
 print white " Name
                          BSSID Signal Noise Channel Type";
 for (x in name)
  {
        #arbitrary strength calc through trial and error... modify as you wish:
        sigstrength = ((rssi[x] - noise[x])*1.5) + ((rssi[x] +90)*1.5);
        if (sigstrength <1) sigstrength=0;
       cStrength = cStrengthLow;
        if(sigstrength>4) cStrength = cStrengthMed;
        if(sigstrength>7) cStrength = cStrengthHi;
        if (age [x]=0) cStrength = cStrengthAged;
        fmt = "%s%-15s %s%0"sigstrength"d "reset erase2end "\n %s %s%-4d %s%-4d %s%-4d %s%2s
       printf fmt, cName,name[x],cStrength,0,x,cSignal,rssi[x],cNoise,noise[x],cChannel, channel
        rssi[x] = "-100 xxxx";
       ln++;
  }
 if (ln ==0)
   print red "No results - Do you have survey capability? \nThis program depends on site_survey
 print erase2end;
}
}
```

## Name-based WOL (wake.sh)

• Enables you to power on a LAN computer by name instead of IP address/MAC, based on DHCP lease table (mandatory).

Usage: /path/to/wake.sh <hostname> (default hostname is desktop)

```
STATION=mm
WOL=/usr/sbin/wol
STATICS=/tmp/udhcpd.statics
DEV=br0
if [ -n "" ]; then
   STATION=
fi
while read LINE
do
   IP=`echo $LINE | awk '{print }'`
  MAC=`echo $LINE | awk '{print }'`
  FOUND=`ip neigh | grep "$IP.*REACHABLE"`
   if [ -z "$FOUND" ]; then
      echo Creating ARP entry for $IP $MAC
      ip neigh add $IP lladdr $MAC dev $DEV nud reachable 2> /dev/null
      ip neigh change $IP lladdr $MAC dev $DEV nud reachable 2> /dev/null
   fi
done < $STATICS
LEASE=`grep "\b$STATION\b$" $STATICS`
if [ -n "$LEASE" ]; then
   IP=`echo $LEASE | awk '{print }'`
  MAC=`echo $LEASE | awk '{print }'`
   $WOL -i $IP $MAC
else
   echo Unable to find \"$STATION\" in DHCP static file $STATICS, please use \" \<hostname\>\"
fi
```

### Name-based WOL with dnsmasq

I updated the above script to work with dnsmasq dhcp reservation.

```
#!/bin/sh
STATION=$1
WOL=/usr/sbin/wol
STATICS=/tmp/dnsmasq.conf
DEV=br0
if [ -z "$STATION" ]; then
   echo "You must specify the host to wake. Usage: $0 <hostname>"
   exit 1
fi
grep 'dhcp-host' $STATICS | while read LINE
do
  IP=`echo $LINE | awk -F, '{print $3}'`
  MAC=`echo $LINE | awk -F[,=] '{print $2}'`
   FOUND=`ip neigh | grep "$IP.*REACHABLE"`
   if [ -z "$FOUND" ]; then
      echo Creating ARP entry for ip $IP and MAC $MAC
      ip neigh add $IP lladdr $MAC dev $DEV nud reachable 2> /dev/null
      ip neigh change $IP lladdr $MAC dev $DEV nud reachable 2> /dev/null
   fi
done
```

```
LEASE=`grep ",$STATION," $STATICS`
if [ -n "$LEASE" ]; then
    NET=`echo $LEASE | awk -F, '{split($3,ip,".");printf("%s.%s.%s.255",ip[1],ip[2],ip[3])}'`
    IP=`echo $LEASE | awk -F, '{print $3}'`
    MAC=`echo $LEASE | awk -F[,=] '{print $2}'`
    echo Waking up $STATION on broadcast $NET at ip $IP and MAC $MAC
    $WOL -i $NET $MAC
else
    echo "Unable to find \"$STATION\" in DHCP static file $STATICS."
    exit 1
fi
```

### **Automatic Connection Repair**

• Pings your default gateway every time and force a DHCP renew if no packets are received. Usage: /path/to/always\_on.sh &

```
#!/bin/sh
INTERVAL=10
PACKETS=1
UDHCPC="udhcpc -i vlan1 -p /var/run/udhcpc.pid -s /tmp/udhcpc"
IFACE=vlan1
ME=`basename $0`
RUNNING=`ps | awk '/'"$ME"'/ {++x}; END {print x+0}'`
if [ "$RUNNING" -gt 3 ]; then
   echo "Another instance of \"$ME\" is running"
   exit 1
fi
while sleep $INTERVAL
do
  TARGET=`ip route | awk '/default via/ {print $3}'`
  RET=`ping -c $PACKETS $TARGET 2> /dev/null | awk '/packets received/ {print $4}'`
   if [ "$RET" -ne "$PACKETS" ]; then
      echo "Ping failed, releasing IP address on $IFACE"
        #send a RELEASE signal
      kill -USR2 `cat /var/run/udhcpc.pid` 2> /dev/null
        #ensure udhcpc is not running
      killall udhcpc 2> /dev/null
      echo "Renewing IP address: $IFACE"
      $UDHCPC
     echo "Waiting 10 s..."
      sleep 10
   else
      echo "Network is up via $TARGET"
   fi
done
```

• The following version will work even on resource-starved Linksys WRT54G v8, which lacks most programs needed by the script above. To use it, just add this code to DD-WRT's startup script using the web interface.

INTERVAL=10

```
while true; do
 while [ \! $gw ]; do
   sleep 30
   route -n >/tmp/routes
   while read dest gw foo; do
     if [ $dest = "0.0.0.0" ]; then
       break
     fi
   done </tmp/routes
 done
 logger "auto-repair: default gateway is $gw"
 while ping -qc 2 $qw >/dev/null ; do
    sleep $INTERVAL
 done
 logger "auto-repair: gateway down, restarting WAN"
 kill -USR1 `cat /var/run/udhcpc.pid`
 unset gw
done &
```

• The following solution is even simpler, but only works on repeaters and clients. To use it, just add this code to DD-WRT's startup script using the web interface.

```
#!/bin/sh
# A startup script that rivals watchdog in repeater or client mode
# When used on a Linksys WRT310N v2 that repeats a dodgy AP,
    we get 7+ day uptimes vs the 3-5h uptimes given by watchdog
#
# (tested with build 15940) (p.s. New builds don't like wland bounced, and it prevents client ass
******
# After a successful ping, wait this long to check again.
# Recommend setting this to the time it takes to grab a cup of coffee
coffee=300
while true; do
  count=0
  until ping -c 2 -W 2 www.google.com; do
     # When we cannot get two consecutive pongs,
         then attempt to restore connection
     #
     if [ $count -eq 3 ]; then
        # If we have tried bouncing wland 3 times,
        #
            then try rebooting the router
        reboot
     fi
     wland
            # bounce wland
     sleep 45 # wait 45 seconds for association
     count=`expr $count + 1`
  done
  sleep $coffee
```

```
done
```

## Modifying \$PATH Manually (path.sh)

• Enables adjustment of paths on a per-use basis (i.e. when you're running a terminal and need the new paths, run this script.).

```
#!/bin/sh
export PATH=$PATH:/mmc/bin:/whatever/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/mmc/lib:/whatever/lib
```

Alternatively, if you want to give priority to you're personally installed applications (i.e. you've installed a more robust version of grep, and want to use it by default), add the new paths before \$PATH and \$LD\_LIBRARY\_PATH, as shown below.

```
#!/bin/sh
export PATH=/mmc/bin:/whatever/bin:$PATH
export LD_LIBRARY_PATH=/mmc/lib:/whatever/lib:$LD_LIBRARY_PATH
```

## View Logfile in Browser without Local Syslogd (log.sh)

• View the last 1000 lines from your router's logfile in your browser without a locally running syslogd (i.e. Kiwi)

#### First Method: Script Generated Live-content

Initial post in German forum: SOLVED: messages (logdatei) formatiert über browser aufrufen)

```
#!/bin/sh
echo '<HTML><HEAD><TITLE>Logfile</TITLE></HEAD>'
echo '<BODY>'<br />nvram get router_name
echo ' Logfile:<br>'
/usr/bin/tail -n 1000 /var/log/messages
echo '</BODY></HTML>'
```

To use this script you first need to enable syslog on your router without stating an IP. Then the log will be saved in /var/log/messages. You can do this under Administration->Services and then scroll down to the "System Log" section. Click "Enable" and leave "Remote Server" empty. After you saved the script under /tmp/www/ as "log.sh" you must mark it as executable with "chmod +x /tmp/www/log.sh". You can do that by saving the following in your startup script:

```
echo -en "#!/bin/sh\necho '<HTML><HEAD><TITLE>Logfile</TITLE></HEAD>'\necho '<BODY>'\nnvram get r
```

To view the log in your browser point it to "http://<routerip>/user/log.sh"

It appears that the above method doesn't work under some versions of v24 as shell scripts need to be created in the cgi-bin folder in order for the webserver to execute them.

If you find the previous startup script doesn't work, try the following:

```
mkdir /tmp/www/cgi-bin
echo -en "#!/bin/sh\necho '<HTML><HEAD><TITLE>Logfile</TITLE></HEAD>'\necho '<BODY>'\nnvram get r
chmod +x /tmp/www/cgi-bin/log.sh
```

and use http://<routerip>/user/cgi-bin/log.sh to access it.

#### Second Method: Static Generated HTML

Note that it is reported that script-generated content will not be delivered by the web server in v24-RC4 and v24-RC5, maybe other versions are affected too (see <u>User-HTML (skript generiert) funzt nicht :(</u> in the German forum). If you just get an empty page if using the first method you may use this workaround:

```
echo -en "#!/bin/sh\nrm /tmp/www/syslog.html\necho '<HTML><HEAD><TITLE>Logfile (Generated: ' >> /
chmod +x /tmp/www/log_gen.sh
```

Save the above code to your startup script and create a cron job for it. To generate a HTML log every 15 minutes you could use this job:

\*/15 \* \* \* \* root /tmp/www/log\_gen.sh

Your router's syslog is now available on http://<routerip>/user/syslog.html and will be updated every 15 minutes (or whatever you've set in the cron job).

### **Speak Your Signal Strength**

I use my WRT in client mode to connect to an access point, but I don't have a particularly good signal quality and I often need to re adjust the position of the WRT and its antenna. Unfortunately my computer is not in sight of the WRT and I had to keep going backwards and forwards from my computer to the WRT making adjustments then checking the signal strength on the screen of my computer. This can take ages to to set up properly, so I decided to get my computer to use the "festival" speech synthesis program to tell me what the current signal level is.

```
#! /bin/bash
# Use "festival" to say out loud how much signal strength we have
# The IP address of the WRT
ip_addr="192.168.1.1"
# The username and password for the WRT
user="root"
pass="admin"
# Tempory file used to hold the data from the WRT
tmp_file=/tmp/wrt.status
echo
echo "The signal level is:-"
echo
echo "The signal level is" | festival --tts
while true ; do
 wget --http-user=$user --http-password=$pass http://$ip_addr/Status_Wireless.live.asp -0 $tmp_f
 signal=`awk -F "'" '/active_wireless/ { print }' $tmp_file`
 echo $signal | awk '{printf"Signal : ""\t";for(;j<;j++)printf"=";printf"\n"}'</pre>
  if [[ -n $signal ]]; then
    echo $signal | festival --tts
```

```
else
echo "Not associated" | festival --tts
fi
done
```

This works by using the same process as the 'Status-->Wireless' page i.e. it gets a chunk of data by wget'ing the Status\_Wireless.live.asp page from the WRT then running awk to get the relevant chunk of data (the signal strength) and then piping that into the festival speech engine.

Now I just run this script and turn up the volume on my computer when I need to move the antenna.

## Small Security Script (Firewall)

```
#!/bin/sh
# Warning! As I don't use Emule or similiar programs I can't guaranty their function.
# If you find a workable solution just add it to this wiki.
# I found testing some of the setting manually that the ipfrag settings will break emule,
# maybe some others too...
#
# Enjoy your enhanced security,
#
# St. Karitzl
# info@user1.walztech.de
# http://daywalker81.de.vu
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/ip_forward
# the following two parametes will break at least emule and are way too low to make sense.
#echo 1024 > /proc/sys/net/ipv4/ipfrag_high_thresh
#echo 512 > /proc/sys/net/ipv4/ipfrag_low_thresh
echo 64000 > /proc/sys/net/ipv4/ipfrag_high_thresh
echo 48000 > /proc/sys/net/ipv4/ipfrag_low_thresh
echo 10 > /proc/sys/net/ipv4/ipfrag_time
echo 5 > /proc/sys/net/ipv4/icmp_ratelimit
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
echo 0 > /proc/sys/net/ipv4/conf/eth1/accept_source_route
echo 0 > /proc/sys/net/ipv4/conf/eth1/accept_redirects
echo 1 > /proc/sys/net/ipv4/conf/eth1/log_martians
echo 10 > /proc/sys/net/ipv4/neigh/eth1/locktime
echo 0 > /proc/sys/net/ipv4/conf/eth1/proxy_arp
echo 50 > /proc/sys/net/ipv4/neigh/eth1/gc_stale_time
# The following entries secure the last bit and provide a
# moderate protection against man-in-the-middle attacks.
#
echo 0 > /proc/sys/net/ipv4/conf/eth1/send_redirects
echo 0 > /proc/sys/net/ipv4/conf/eth1/secure_redirects
echo 1 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
echo 5 > /proc/sys/net/ipv4/igmp_max_memberships
echo 2 > /proc/sys/net/ipv4/igmp_max_msf
```

echo 1024 > /proc/sys/net/ipv4/tcp\_max\_orphans echo 2 > /proc/sys/net/ipv4/tcp\_syn\_retries echo 2 > /proc/sys/net/ipv4/tcp\_synack\_retries echo 1 > /proc/sys/net/ipv4/tcp\_abort\_on\_overflow echo 10 > /proc/sys/net/ipv4/tcp\_fin\_timeout echo 0 > /proc/sys/net/ipv4/route/redirect\_number echo 1 > /proc/sys/net/ipv4/conf/all/rp\_filter echo 1 > /proc/sys/net/ipv4/conf/eth1/rp\_filter echo 1 > /proc/sys/net/ipv4/tcp\_syncookies echo 0 > /proc/sys/net/ipv4/conf/all/accept\_source\_route echo 61 > /proc/sys/net/ipv4/ip\_default\_ttl # DoS protection by tweaking the timeouts echo "1800" > /proc/sys/net/ipv4/tcp\_keepalive\_time echo "0" > /proc/sys/net/ipv4/tcp\_window\_scaling echo "0" > /proc/sys/net/ipv4/tcp\_sack # We pretend to be a Checkpoint firewall on Windows XP echo 4096 87380 4194304 >/proc/sys/net/ipv4/tcp\_rmem echo 4096 87380 4194304 >/proc/sys/net/ipv4/tcp\_wmem # Check network overload (explicit congestion notification) echo 1 > /proc/sys/net/ipv4/tcp\_ecn # Change port range for outgoing traffic echo "30000 60000" > /proc/sys/net/ipv4/ip\_local\_port\_range # Change default queue size # Modified for DD-WRT because of missing proc entries echo 4096 > /proc/sys/net/ipv4/ip\_conntrack\_max # LED signal feedback when script ends sleep 1 gpio enable 3 sleep 1 gpio disable 3 sleep 1 gpio enable 3 sleep 1 gpio disable 2 sleep 1 gpio enable 2 sleep 1 gpio disable 2 # If you'd like to disable the web interface uncomment # the following line #killall httpd

Attention, you might have to change eth1 to the actual WAN (external) interface.

Installation is pretty simple:

- 1. Log on to your WRT
- 2. type cd /jffs
- 3. type vi sec.sh (or any other name) and enter the script
- 4. Connect to your WRT via web browser, page Administration: Commands
- 5. Enter the script name (sec.sh) into the command field

6. Click on "Save Startup"

7. Reboot router

As a simple test try to ping your router. You should get no response otherwise you have to find the error.

## Secure remote management for a <u>WAP</u> (Firewall scripts)

Regards to the help of phusiOn dd-wrt guru and of HP from ubuntu-fr

This requires a recent (>12533) build to prevent milw0rm and to have the Disable "Allow any remote IP" feature. Also requires you have set the necessary port forwards in the gateways(s) on the path.

```
iptables -I INPUT -p icmp --icmp-type echo-request -m state --state NEW -j ACCEPT
iptables -I INPUT -i lo -j ACCEPT
IP_LAN=`ifconfig br0 | grep inet | cut -d: -f2 | cut -d' ' -f1`
MSK=`ifconfig br0 | grep inet | cut -d: -f4`
iptables -I INPUT -p tcp -s $IP_LAN/$MSK -d $IP_LAN -m multiport --dports `nvram get sshd_port`,`
iptables -I INPUT -p tcp -s `nvram get remote_ip | awk '{print $1}'` -d $IP_LAN -m multiport --dp
# iptables -I INPUT -p udp --sport 68 --dport 67 -m state --state NEW -j ACCEPT # uncomment if yo
iptables -I INPUT -d $IP_LAN -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -j logreject
```

This allows the lowest (first) IP address set in the "Allow any remote IP" feature to connect to the https and ssh servers in the <u>WAP</u> (you just need the password and/or the private key) ;) . In addition, all LAN ip are allowed to do the same, plus classic http. Althought the "Allow any remote IP" feature doesn't work at this time when the router is set as a <u>WAP</u> (LAN-LAN link to the gateway, so no routing and WAN disabled, dhcp off and other stuff), this convenient script will use the first "remote\_ip" you set in the GUI (whether the feature is enabled or not, as long as you stored at least one IP) and will follow the changes you could do to its static lan ip/netmask and ssh server port. Now you have full benefit of the GUI from a remote static IP and can leave the forwards enabled on the path.

EDIT 2011/06/25: even with elder releases, the remote IP can be hardcoded in the script, replacing `*nvram get remote\_ip* | *awk '{print }*'` by the actual IP.

--Bib 14:03, 17 May 2010 (CEST)

EDIT 2014/03/04: I slightly changed this script to make it work when the <u>WAP's LAN IP is set through dhcp</u> instead of static (but I believe it is not possible at all in this case to have the dd-wrt dhcp server running as *if it worked* it would serve an IP to himself <u>YPIO YPIO</u>):

```
iptables -I INPUT -p icmp --icmp-type echo-request -m state --state NEW -j ACCEPT
iptables -I INPUT -i lo -j ACCEPT
iptables -I INPUT -p udp --sport67 --dport 68 -m state --state NEW -j ACCEPT
IP_LAN=`ifconfig br0 | grep inet | cut -d: -f2 | cut -d' ' -f1`
MSK=`ifconfig br0 | grep inet | cut -d: -f4`
iptables -I INPUT -p tcp -s $IP_LAN/$MSK -d $IP_LAN -m multiport --dports `nvram get sshd_port`,`
iptables -I INPUT -p tcp -s `nvram get remote_ip | awk '{print $1}'` -d $IP_LAN -m multiport --dp
iptables -I INPUT -d $IP_LAN -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -j logreject
```

Note: 2nd line as per iptables command wiki, but I'm not sure this is still required here or when you set a

Secure remote management for a WAP (Firewall scripts)

static lease for your WAP in the DCHP server machine/box. This allows remote ssh/https. With no need to change the script, you can also occasionally enable lan-only simple http access from within the GUI for some operations (firmware backup/restore/upgrade): in secured management page, just temporarily check the HTTP checkbox, apply and reconnect without https. Or do this from ssh:

```
nvram set http_enable=1 && nvram commit && killall httpd && httpd -S
```

and when you finished the job, set it back:

nvram set http\_enable=0 && nvram commit && killall httpd && httpd -S

--<u>Bib</u>

### **Directory Listing for DD-WRT Micro**

Since the Micro version of DD-WRT doesn't provide a ls command, here is a very simple script to list directory contents

```
#!/bin/sh
files=`echo *`
for x in $files; do
    if [ -d $x ]; then
        echo -n "$x/ "
    else
        echo -n "$x "
    fi
    done
    echo
```

See the Telnet/SSH and the Command Line Talk page for other variants.

### **Reset Wireless Radio**

This script solves an intermittent problem on my NetGear WNDR3300 wireless N radio. Every few hours, the wireless N radio stops broadcasting and cannot be seen by wireless clients. Bringing the wireless interface down and then back up resolves the issue. This script pings a wireless client, in my case, a WET610N wireless bridge that should always remain up and only connects to the wireless N radio. If the ping fails twice within a given time, it brings the interface down and then back up.

To get this to work on Atheros hardware, replace the two 'wl - i'... comands with 'ifconfig'... (e.g., this fixes wifi dropouts on my Buffalo WZR-HP-G300NH)

```
#!/bin/sh
# This script solves an intermitent problem on my
# NetGear WNDR3300 wireless N radio. Every few
# hours, the wireless N radio stops broadcasting
# and cannot be seen by wireless clients. Bringing
# the wireless interface down and then back up
```

# resolves the issue. This script pings a # wireless client, in my case, a WET610N wireless # bridge that should always remain up and only # connects to the wireless N radio. If the # ping fails twice within a given time, it # brings the interface down and then back up. # A wireless client that should always be up CLIENT IP=192.168.35.250 # Wireless interface that disappears INTERFACE=`nvram get wl0\_ifname` # seconds to wait after failed ping to try again FAIL\_AGAIN=10 # seconds between checks CHECK\_EVERY=60 # after cycling, wait this many seconds AFTER\_CYCLE=360 # Client must be up before starting main loop while true do if ping -c 1 \${CLIENT\_IP} >/dev/null then echo "\${CLIENT\_UP} ok - begining main loop" break fi done # main script while sleep \${CHECK\_EVERY} do if ping -c 1 \${CLIENT\_IP} >/dev/null then echo "\${CLIENT\_IP} ok" else echo "\${CLIENT\_IP} dropped one" sleep \${FAIL\_AGAIN} if ! ping -c 1 \${CLIENT\_IP} >/dev/null then echo "\${CLIENT\_IP} dropped two, sending restarting \${INTERFACE}" # on Atheros hardware instead use: ifconfig \${INTERFACE} down wl -i \${INTERFACE} down sleep 3 # on Atheros hardware instead use: ifconfig \${INTERFACE} up wl -i \${INTERFACE} up sleep \${AFTER\_CYCLE} fi fi done 2>&1

### **Display Connection Counts Per IP**

#### For local IP's

for i in `grep 0x /proc/net/arp | cut -d ' ' -f1`; do echo "\$i connection count: \$(grep -c \$i /pr

#### For all IP's

for i in `grep 0x /proc/net/arp | cut -d ' ' -f1`; do echo "\$i connection count: \$(grep -c \$i /pr

## L2TP Fix for HOT users (obsolete)

Script for HOT users at Israel who can't connect using L2TP because DD-WRT default setting for "refuse pap=yes" while HOT setting is the opposite. Fix was risen by aviad\_ra @ <a href="http://www.dd-wrt.com/phpBB2/viewtopic.php?t=17002&postdays=0&postorder=asc&highlight=12tp&start=15">http://www.dd-wrt.com/phpB2/viewtopic.php?t=17002&postdays=0&postorder=asc&highlight=12tp&start=15</a>

This is a cleaned up version of the same script. Add the script as a startup script.

## **Email Bandwidth Usage Daily**

This script will email you with the previous day's and total month's bandwidth usage. Edit it with your sendmail SMTP info. <u>More in-depth examples</u> <u>Original Forum Topic</u>

```
#!/bin/sh
fnc_mail() {
    subj="$1"
    msg="$2"
    if [ -z "$3" -o "$(dirname $3)" = "." ]; then logfile="/tmp/lastmail.log"; else logfile="$3";
    sendmail -S"smtp.comcast.net" -f"sender@comcast.net" -F"DD-WRT" -d"comcast.net" -s"$subj" -m"$
}
#Parse the previous day and monthly totals and send an email
aff="aff" #keyword workaround
yday=$(date -D %s -d $(( $(date +%s) - 86400)) +%d)
ymon=$(date -D %s -d $(( $(date +%s) - 86400)) +tr$aff-%m-%Y)
if [ $(date +%d) -eq 1 ]; then monmsg="Last Month"; else monmsg="Month to Date"; fi
```

msg=\$(nvram get \$ymon | awk '{print \$'\$yday', \$NF}' | sed 's/\([^:]\*\):\([^ ]\*\) \[\([^:]\*\):\([^
fnc\_mail "Bandwidth Report" "\$msg" "/tmp/bwmail.log"

### Update Allowed Remote IP Range by DNS lookup

If you are allowing Remote Access to manage your router from a remote location, this script will update the IP address range via a DNS lookup, in the event that the IP address assigned to the remote location changes. For home users, an account with a DDNS service such as DynDNS will work.

```
#Variables
#
# use a 10 minute interval
INTERVAL=600
#set DNS entry to check
REMOTE_DNS=*CHANGE TO YOUR DNS*
while sleep $INTERVAL
do
# Set remote IP variable
REMOTE_IP=`ping -c 1 $REMOTE_DNS | grep "^PING" | cut -f3 -d" " | cut -f2 -d"(" | cut -f1 -d")"`
#Set ending octet variable to be one more than the last octet of the remote address
let ENDING_OCTET=`echo $REMOTE_IP | cut -f4 -d"."`+1
#determine the current IP address that's used for remote management
CURRENT_ADMIN_IP=`nvram get remote_ip | awk '{print $1}'
### If the remote IP has changed, then change the admim IP setting
if [ "$CURRENT_ADMIN_IP" != "$REMOTE_IP" ]; then
       #configure WAP to allow this IP address to connect remotely
       nvram set remote_ip="$REMOTE_IP $ENDING_OCTET"
       #save nvram settings
       nvram commit > /dev/null 2>&1
fi
done
```

### End of script

## **DynDNS Updates Using Curl (with HTTPS/SSL Support)**

DD-WRT's included inadyn client does not support HTTPS updates. This means your credentials are sent in cleartext across the Internet whenever an update is made. Using optware and curl, we can make secure updates over HTTPS instead of HTTP

#!/bin/sh # DynDNS Updater Script, with HTTPS/SSL Support # This example uses NameCheap's HTTPS/SSL URL update method to update the IP for hostname.example # Edit the settings section to match your DynDNS provider, including the DDNS URL which will be u # This script requires curl and its CA Bundle to be installed in /opt/usr/bin, see the DynDNS art # alfer@fusebin.com ### SETTINGS SECTION ### UPDATE INTERVAL=300 DDNS\_DOMAIN=example.com DDNS HOSTNAME=hostname DDNS\_USERNAME=username DDNS\_PASSWORD=password123 DDNS\_URL="https://dynamicdns.park-your-domain.com/update?domain=\$DDNS\_DOMAIN&password=\$DDNS\_PASSW LOG\_FILE=/var/log/dyndns.log CURL\_LOG\_FILE=/var/log/curl.log CURL\_LOCATION=/opt/usr/bin/curl CA\_BUNDLE=/opt/usr/bin/ca-bundle.crt ### END SETTINGS SECTION while sleep \$UPDATE\_INTERVAL do CURRENT\_TIME=`date` CURRENT\_WAN\_IP=`nvram get wan\_ipaddr` LAST\_WAN\_IP=`nvram get wan\_ipaddr\_last` DNS\_IP=`nslookup \$DDNS\_HOSTNAME.\$DDNS\_DOMAIN 8.8.8.8 | grep Address | tail -n 1 | awk '{p DDNS\_UPDATE\_COMMAND="\$CURL\_LOCATION -0 \$CURL\_LOG\_FILE --cacert \$CA\_BUNDLE \$DDNS\_URL" # Check if IP Address has changed from last locally stored IP if [ \$CURRENT\_WAN\_IP != \$LAST\_WAN\_IP ]; then echo "\$CURRENT\_TIME: IP has changed from \$LAST\_WAN\_IP to \$CURRENT\_WAN\_IP" >> \$LOG # Update the last locally stored IP nvram set wan\_ipaddr\_last=\$CURRENT\_WAN\_IP nvram commit \$DDNS\_UPDATE\_COMMAND echo "\$CURRENT\_TIME: Updated DynDNS Service, server sent response:" >> \$LOG\_FILE echo -e "\n" >> \$LOG\_FILE cat \$CURL\_LOG\_FILE >> \$LOG\_FILE echo -e "\n" >> \$LOG\_FILE rm \$CURL\_LOG\_FILE continue # Check if IP Address has changed from what DNS is reporting elif [ \$CURRENT\_WAN\_IP != \$DNS\_IP ]; then echo "\$CURRENT\_TIME: DNS IP not up to date, currently: \$DNS\_IP, but our IP is \$CU \$DDNS\_UPDATE\_COMMAND echo "\$CURRENT\_TIME: Updated DynDNS Service, server sent response:" >> \$LOG\_FILE echo -e "\n" >> \$LOG\_FILE cat \$CURL\_LOG\_FILE >> \$LOG\_FILE echo -e "\n" >> \$LOG\_FILE rm \$CURL\_LOG\_FILE fi done

**Other Useful Misc. Scripts** 

These scripts will be executed through either SSH or Telnet. Once logged into you router via one of these methods, the following are quite useful:

df - Displays mounted external devices, such as USB storage. Use df -h to see the actual size va mount - Mounts an external device such as a USB HDD (Example: mount /dev/scsi/host0/bus0/target0 mount --bind - Binds an old directory to a new directory (Example: mount --bind /tmp/mnt/tmp) - I umount - Unmounts the external device top - Displays all running processes including the CPU usage, Memory Usage and so on dmesg | more - Will display Router information one segment at a time (terminal window size) and k netstat -an - Will display all incoming/outgoing connections **ps** - Will display all running processes - Use **ps** --help to display more options. free - Displays current memory statistics, and be used with extensions (Example: free -1) - Use f cat - Very, very useful and can display all input/output messages from the router" (Example: cat nvram show - Displays every nvram variable and available nvram space nvram set - Set the value of an nvram variable (Example: nvram set rc\_firewall="echo hello") nvram unset - Deletes an nvram variable, not only its content (Example: nvram unset traff-13-2110 nvram commit - Commits nvram variables to the flash chip reboot - Reboots the router, and must be used after an nvram commit cd /jffs - Changes to the jffs directory cd /mnt - Changes to the mounted directory cd /opt - Changes to the /opt directory

#### These commands can also be be bound. (Example: cd /mnt/opt/etc)

NOTE: The following commands are used when the above commands have been executed

- mkdir Makes a directory, in which of course you need to specify (Example: mkdir /mnt/jffs) To m
  rmdir Removes the specified directory
- rm Removes a directory (Example: cd /opt then rm -rf \*) -> Removes the/opt directory
- killall kill a specific process (Example: killall syslogd)
- sleep Causes the process to delay startup for specified time (Example: sleep 20 && /opt/bin/bus

chown - Change the owner and/or group of each FILE to OWNER and/or GROUP - Type chown --help for chmod - Allows permissions changing on directories (Example: chmod +x /jffs/etc/config/) - The +x stopservice - Stops the specified service (Example: stopservice upnp) - Replace stop with start t

# See Also

LED Scripts Startup Scripts Script Examples