

Scripte

- [check_interfaces.sh](#)
- [check_ldap_response.sh](#)
- [pingall.sh](#)
- [check_vips.sh](#)
- [count-packets.sh](#)
- [cisco.sh](#)
- [vpn-keepalive.sh](#)
- [ssh-break.sh](#)
- [signature.sh](#)
- [wake.sh](#)
- [do-cmd.sh](#)

check_interfaces.sh

```
#!/bin/sh

# checkt ob alle Interface up sind und Link haben und liefert einen fuer Nagios gueltigen
Return-Wert

CARDS="dev0 eth1 eth2 eth3"

DEBUG=0
MESG=""
INTOK="OK:"

for card in $CARDS
do
    if [ $DEBUG -gt 0 ]
    then
        echo -n "Testing $card "
    fi
    ret=$(ethtool -t $card online > /dev/null 2>&1 ; echo $?)
    if [ $ret -gt 0 ]
    then
        echo "CRITICAL: Interface $card failed!"
        exit 2
    else
        INTOK="$INTOK $card"
    fi
done
echo "$INTOK operating normally"
exit 0
```

check_ldap_response.sh

zuerst muss mit dem Parameter -g ein Diff-File erzeugt werden, dann können mit dem Parameter -f die Antworten der LDAP-Server mit dem File verglichen werden.

```
#!/bin/sh

# 2005 Oliver Voelker <code@magenbrot.net>
# checkt die LDAP-Server auf korrekte Antworten und Synchronisation

# usage:
# check_ldap_response.sh -H <host> [-p <port>] -b <base_dn> -D <binddn> -P <bindpassword> -f
<checkfile> -v
#
# generate check-file:
# check_ldap_response.sh -H <host> [-p <port>] -b <base_dn> -D <binddn> -P <bindpassword> -g
<checkfile> -v

LDAPSEARCH=`which ldapsearch`

PORT="389"
LDAPLOOKUP="uid=testuser"
TMP=/tmp/ldap.$$
DIFF=$(dirname $0)/ldap-checkfile

while getopts "H:p:b:D:P:g:f:v" flag
do
    #echo "$flag" $OPTIND $OPTARG
    case "$flag" in
        H) HOST=$OPTARG
            ;;
        p) PORT=$OPTARG
            ;;
        b) BASE=$OPTARG
            ;;
        D) BINDDN=$OPTARG
            ;;
        P) BINDPW=$OPTARG
            ;;
    esac
done
```

```

g) DIFF=$(dirname $0)/ldap-$OPTARG
  echo "lege Vergleichsfile $DIFF an..."
  $LDAPSEARCH -h $HOST -p $PORT -b $BASE -D $BINDDN $LDAPLOOKUP -x -w $BINDPW > $DIFF
  exit 0
  ;;
f) DIFF=$(dirname $0)/ldap-$OPTARG
  ;;
v) $LDAPSEARCH -h $HOST -p $PORT -b $BASE -D $BINDDN $LDAPLOOKUP -x -w $BINDPW > $TMP
  diff -y $TMP $DIFF
  ;;
*) echo "Ungueltiges Kommandozeilenargument"
  exit 1
  ;;
esac
done

if [ -z $HOST ] || [ -z $BASE ] || [ -z $BINDDN ] || [ -z $BINDPW ]; then
  echo "Fehlendes Argument!"
  echo "usage:"
  echo "check_ldap_response.sh -H <host> [-p <port>] -b <base_dn> -D <binddn> -P
<bindpassword> -f <checkfile>"
  echo "generate checkfile:"
  echo "check_ldap_response.sh -H <host> [-p <port>] -b <base_dn> -D <binddn> -P
<bindpassword> -g <checkfile>"
  exit 2
fi

if [ -z $LDAPSEARCH ]; then
  echo "ldapsearch (openldap-clients) nicht gefunden, bitte installieren."
  exit 2
fi

if [ ! -r $DIFF ]; then
  echo "Vergleichsfile $DIFF existiert nicht oder kann nicht gelesen werden, bitte mit
parameter -g aufrufen!"
  exit 2
fi

$LDAPSEARCH -h $HOST -p $PORT -b $BASE -D $BINDDN $LDAPLOOKUP -x -w $BINDPW > $TMP

```

```
diff $TMP $DIFF > /dev/null 2>&1
```

```
if [ $? != "0" ]; then
```

```
    echo "LDAP Server nicht synchron bzw. Antwort fehlerhaft"
```

```
    exit 2
```

```
fi
```

```
rm $TMP
```

```
echo "LDAP Server ist synchron"
```

pingall.sh

```
#!/bin/sh

SERVERLIST=serverlist.txt

if [ -e $SERVERLIST ]; then
  for SERVER in `cat $SERVERLIST`; do
    trap 'exit 0' 2
    ping -c2 $SERVER
    echo
  done
fi
```

check_vips.sh

```
#!/bin/sh

# quick n very dirty check

# wieviele IPs sollte ich haben?
SHOULD=10

VIPs=`/sbin/ip a | grep "inet " | wc -l | awk -F" " '{ print $1 }'`

if [ ! "$VIPs" -eq "$SHOULD" ]; then
    echo "CRITICAL: owned IPs: $VIPs - should be: $SHOULD"
    exit 2
else
    echo "OK: owned IPs: $VIPs"
    exit 0
fi
```

count-packets.sh

```
#!/bin/bash
#
# 2008 Oliver Voelker <info(at)ovtec.it>
#
# Pakete pro Sekunde fuer ein bestimmtes Interface ermitteln
#

if [ -z $1 ]; then
    echo "Usage: $0 <interface> [<time>]"
    echo "Default measure time is 10 seconds"
    exit 0
fi

if [ -z $2 ]; then
    TIME=10
else
    TIME=$2
fi

DEV=$1

RUN=`ifconfig $DEV | grep "X packets"`
if [ $? -ne 0 ]; then
    echo "Fehler!"
    exit 1
fi

RX1=`echo $RUN | cut -d" " -f2 | cut -d":" -f2`
TX1=`echo $RUN | cut -d" " -f8 | cut -d":" -f2`

#echo "RX-1: $RX1"
#echo "TX-1: $TX1"

sleep $TIME

RUN=`ifconfig $DEV | grep "X packets"`
```

```
if [ $? -ne 0 ]; then
    echo "Fehler!"
    exit 1
fi

RX2=`echo $RUN | cut -d" " -f2 | cut -d":" -f2`
TX2=`echo $RUN | cut -d" " -f8 | cut -d":" -f2`

#echo "RX-2: $RX2"
#echo "TX-2: $TX2"

RX=$(( $RX2 - $RX1 ))
TX=$(( $TX2 - $TX1 ))
RXps=$(( $RX / $TIME ))
TXps=$(( $TX / $TIME ))

echo "RX in $TIME seconds: $RX (RX per second: $RXps)"
echo "TX in $TIME seconds: $TX (TX per second: $TXps)"
```

cisco.sh

Dieses Script ermöglicht einen automatischen Login auf Cisco-Switches inkl. automatischem „enable“ (sofern die Kennwörter gleich sind):

```
#!/bin/sh

# Passwort
PASS="passwort"
PROGDIR=`dirname $0`

rm -f $PROGDIR/login.expect
touch $PROGDIR/login.expect
chmod u+x $PROGDIR/login.expect

cat << EOF > $PROGDIR/login.expect
#!/usr/bin/expect
spawn telnet $1
expect {
    "Password:" {
        send "$PASS\r"
        exp_continue
    }
    "Name:" {
        send "enable\r"
        exp_continue
    }
    "#" {
        interact
    }
    "% Bad passwords" {
        send_user "invalid password or account\n"
        exit
    }
    timeout {
        send_user "connection timed out\n"
        exit
    }
}
eof {
```

```
send_user "connection to $host failed: $expect_out(buffer)"
exit
}
}
EOF

$PROGDIR/login.expect
rm -f $PROGDIR/login.expect
```

Aufruf mit „cisco.sh cat1.mydomain.de“ oder via IP.

vpn-keepalive.sh

```
#!/bin/bash
# keepalive for ipsec
# 2007 Oliver Voelker <info(at)ovtec.it>

failmax=3 # beim dritten Fehler restarten
keepalive=30 # alle $keepalive Sekunden testen
maxage=120 # maximales Alter der Checkdatei in Sekunden
nextrestart=3600 # nach einem neustart erst wieder in X sekunden probieren

CHECKFILE="/tmp/keep-alive" # Dieses File muss minuetlich durch die VPN-Gegenseite erzeugt
werden, z.B. durch einen Cronjob: "ssh user@bla.de -C touch /tmp/keep-alive"
TMPFILE="/tmp/vpntest-$$"
ADMIN="admin@bla.de" # wird bei Stoerungen informiert
TUNNEL="ipsec-tunnel" # Tunnelname aus ipsec.conf

# do not edit anything beyond this point!

fail=0

MESSAGE=""

function tunnelrestart () {
    MESSAGE="Maxfail ($failmax) reached: restarting tunnel $TUNNEL (age of checkfile $DIFF
seconds)"
    logger -p local2.info -t TUNNEL "$MESSAGE"
    /usr/sbin/ipsec auto --down $TUNNEL
    sleep 5
    /usr/sbin/ipsec auto --up $TUNNEL
    echo "tunnel $TUNNEL on `hostname -f` was restarted, because checkfile $CHECKFILE was too
old. Please check!" | mail -s "VPN-Problem on `hostname -f`!" $ADMIN
    touch $TMPFILE
    sleep 120
}

while (true); do
    CHECK=`stat -c"%Y" $CHECKFILE`
```

```

NOW=`date +%s`
DIFF=`echo $NOW - $CHECK | bc`

if [ "$DIFF" -lt "$maxage" ]; then
    MESSAGE="Tunnel $TUNNEL OK (age of checkfile $DIFF seconds)"
    logger -p local2.info -t TUNNEL "$MESSAGE"
    fail=0
else
    fail=`echo $fail+1|bc`
    MESSAGE="Tunnel $TUNNEL DOWN: $fail (age of checkfile $DIFF seconds with maxage of
$maxage)"
    logger -p local2.info -t TUNNEL "$MESSAGE"
fi

if [ "$fail" -ge "$failmax" ] ; then

    if [ -f $TMPFILE ]; then
        ATMP=`stat -c"%Y" $TMPFILE`
        SSLR=`echo $NOW - $ATMP | bc` # seconds since last restart
        if [ "$SSLR" -ge "$nextrestart" ]; then
            rm -f $TMPFILE
            tunnelrestart
            fail=0
        else
            MESSAGE="Maxfail ($failmax) reached, but tunnel was already restarted $SSLR seconds
ago. Only one restart per $nextrestart seconds."
            logger -p local2.info -t TUNNEL "$MESSAGE"
            echo "tunnel $TUNNEL should have been restarted, but this already happened $SSLR
seconds ago. Please check!" | mail -s "BIG VPN-Problem on vpn.meinserver.de!" $ADMIN
        fi
    else
        tunnelrestart
    fi
fi

sleep $keepalive
done

```

ssh-break.sh

```
#!/bin/bash

# scannt in diesem Beispiel den Netzwerkbereich 192.168.10.1 bis 192.168.20.255 via SSH mit
dem User root und dem Passwort <enter password to check for>
# (<enter password to check for> durch gewuenshtes Passwort ersetzen!)

# von
A1="192."
B1="168."
C1="10."
D1="1"

# bis
A2="192."
B2="168."
C2="20."
D2="255"

# Passwort
PASS="<enter password to check for>"

PROGDIR=`dirname $0`

rm -f $PROGDIR/login.expect
touch $PROGDIR/login.expect
chmod u+x $PROGDIR/login.expect

for a in `seq $A1 $A2`
do
  for b in `seq $B1 $B2`
  do
    for c in `seq $C1 $C2`
    do
      for d in `seq $D1 $D2`
      do
        trap 'exit 0' 2
```

```
        echo "Teste $a.$b.$c.$d:"

cat << EOF > $PROGDIR/login.expect
#!/usr/bin/expect
spawn ssh -o PubkeyAuthentication=no -o ConnectTimeout=1 -o NumberOfPasswordPrompts=1
root@$a.$b.$c.$d "uptime"
expect {
    password: {
        sleep 1
        send "$PASS\r"
        exp_continue
    } "connecting (yes/no)?" {
        send "yes\r"
        exp_continue
    } incorrect {
        send_user "invalid password or account\n"
        exit
    } timeout {
        send_user "connection timed out\n"
        exit
    } eof {
        exit
    }
}
EOF

        $PROGDIR/login.expect
        echo -e
"#####"
        done
        done
        done
done

rm -f $PROGDIR/login.expect
```

signature.sh

```
#!/bin/sh

echo
cat ~/.signature
echo
echo "-----"
echo
/usr/bin/fortune
```

wake.sh

Dieses Script holt einen Rechner/Server/HTPC aus dem Schlaf (sofern Wake On LAN (WOL) im BIOS aktiviert ist) und teilt einem mit, wenn die Maschine eine Netzwerkverbindung hat. Im Beispiel müsst Ihr einfach die Hostnamen und MAC-Adressen durch Eure eigenen ersetzen.

```
#!/bin/sh
#
# Wake On LAN Script
# 2010 Oliver Voelker <info@ovtec.it>
#

case "$1" in
    brot)
        echo "waking up brot"
        wakeonlan 00:1f:c6:0b:9e:3e
        while ( ! ping -c 1 brot.magenbrot.net 2>&1 >/dev/null ); do
            echo -n .
            sleep 1
        done
        ;;
    htpc|*)
        # htpc und default
        echo "waking up htpc"
        wakeonlan 00:01:2e:2b:91:6e
        while ( ! ping -c 1 htpc.magenbrot.net 2>&1 >/dev/null ); do
            echo -n .
            sleep 1
        done
        ;;
esac
echo -e "\nbehold! $1 has awoken!"
```

do-cmd.sh

uralt... Heutzutage nimmt dafür lieber ansible oder ähnliches :)

```
#!/bin/bash
#
# 2006 Oliver Voelker <info(at)ovtec.it>
#

SERVERLIST=serverlist.txt
COMMAND=$*
REPORT=/tmp/report.$$
NOTIFY="scripts@magenbrot.net"
DATE=`date +%d.%m.%y-%H%M`

PROGDIR=`dirname $0`
cd $PROGDIR

if [ ! -f $SERVERLIST ]; then
    echo "$SERVERLIST nicht gefunden."
    exit 1
fi

if [ -z "$1" ]; then
    echo "Usage '$0 <remotecmd>'"
    echo "Executes <remotecmd> on all servers in $SERVERLIST"
    echo
    echo "-check-update          checks for updates on the servers"
    echo "-update                installs available updates"
    echo
    echo "notifies are mailed to $NOTIFY"
    exit 0
fi

echo "Scriptlauf am `date +%c`" > $REPORT
echo "Scriptlauf am `date +%c` geloggt nach $REPORT"
echo "-----" >>
$REPORT
```

```

if [ -e $SERVERLIST ]; then
    echo "-----"
    exec 5<&0
    cat $SERVERLIST | while read LINE; do
        trap 'exit 0' 2
        if [ "`echo $LINE | head -c1`" != "#" ] && [ "`echo $LINE | head -c1`" != "" ]; then
            #echo $LINE | awk -F" - " '{printf "Server: %s / Distri: %s / Updater: %s / Comment:
%s\n", $1, $2, $3, $4}'
            SERVER=`echo $LINE | awk -F" - " '{print $1}'`
            DISTRI=`echo $LINE | awk -F" - " '{print $2}'`
            UPDATER=`echo $LINE | awk -F" - " '{print $3}'`
            COMMENT=`echo $LINE | awk -F" - " '{print $4}'`
            if [ -z "$SERVER" ] || [ -z "$DISTRI" ] || [ -z "$UPDATER" ] || [ -z "$COMMENT" ]; then
                echo "$SERVERLIST hat das falsche Format in Zeile $LINE"
                echo "$SERVERLIST hat das falsche Format in Zeile $LINE" >> $REPORT
                continue
            fi
            echo "Executing '$COMMAND' on $SERVER ($DISTRI / $COMMENT)..."
            echo "Executing '$COMMAND' on $SERVER ($DISTRI / $COMMENT)..." >> $REPORT
            case "$COMMAND" in
                "-check-update")
                    case "$UPDATER" in
                        "yum")
                            ssh $SERVER "sudo su root -c 'yum check-update'" >> $REPORT 2>&1 <&5
                            ;;
                        "apt")
                            ssh $SERVER "sudo su root -c 'apt-get -qq update'" >> $REPORT 2>&1 <&5
                            ssh $SERVER "sudo su root -c 'apt-get -q -u -s upgrade'" >> $REPORT 2>&1 <&5
                            ;;
                        "up2date")
                            ssh $SERVER "sudo su root -c 'up2date --dry-run'" >> $REPORT 2>&1 <&5
                            ;;
                        *)
                            echo "kein gueltiger updater erkannt"
                            ;;
                    esac
                ;;
                "-update")
                    case "$UPDATER" in
                        "yum")
                            ssh $SERVER "sudo su root -c 'yum -y update'" >> $REPORT 2>&1 <&5

```

```

        ;;
    "apt")
        ssh $SERVER "sudo su root -c 'apt-get -y upgrade'" >> $REPORT 2>&1 <&5
        ;;
    "up2date")
        # wenn auch Kernel-Updates mit installiert werden sollen, muss
        # up2date um den Parameter -f ergaenzt werden
        ssh $SERVER "sudo su root -c 'up2date -u'" >> $REPORT 2>&1 <&5
        ;;
    *)
        echo "kein gueltiger updater erkannt"
        ;;
esac
;;
*)
    ssh $SERVER "sudo su root -c '$COMMAND'" >> $REPORT 2>&1 <&5
    ;;
esac
echo "-----"
>> $REPORT
    fi
done
exec 5<&-
fi

echo "-----"
echo "done... notifying $NOTIFY"
mail -s "updater-report" $NOTIFY < $REPORT
cp $REPORT `dirname $0`/report-$DATE.log
echo "Bericht gespeichert unter: `dirname $0`/report-$DATE.log"
echo "-----"

rm -f $REPORT

```