chage-omfg

```
#!/bin/bash

## ----------------------------------------------------------------------------------------------------
--
## check_password_age.sh
## ----------------------------------------------------------------------------------------------------
--
##
## Programmbeschreibung:
## Skript um das Alter des Passworts von root zu ermitteln und eine E-Mail abzusetzen falls das Alter
## einen konfigurierten Wert überschreitet.

## ----------------------------------------------------------------------------------------------------
--
## Initialisierung
## ----------------------------------------------------------------------------------------------------
--

# Debugging Option
# set -x

# /etc/profile sourcen
source /etc/profile

# Systemsprache explizit setzen um zu erzwingen, dass die Ausgabe auf Englisch erfolgt.
export LANG="en_US.UTF-8"


## ----------------------------------------------------------------------------------------------------
--
## Konfiguration des Skripts
## ----------------------------------------------------------------------------------------------------
--

MAXPASSWORDAGE=279;                                     # Maximales Alter des root-Passworts in Tagen

# Pfad zur Userliste
USERLIST="";                                            # Nicht editieren
PATHTOUSERLIST="/root/scripts/chage/data/";             # Pfad zur Datei, die die User zur Ueberpruefung
beinhaltet
USERLISTFILE="users.txt";                               # Dateiname der Datei, die die Userliste
enthaelt

# Mailversand
MAILSENDER="email@domain.com";                          # Absenderadresse
MAILRECEIVER="email@localhost";                         # Empfaengeradresse
SENDMAILIFNOOP="yes";                                   # Soll eine Mail gesendet werden wenn kein
Fehler auftrat (yes|no)
SENDMAILIFERROR="yes";                                  # Soll eine Mail gesendet werden wenn ein Fehler
aufgetreten ist (yes|no)
SENDMAILIFUPDATE="yes";                                 # Soll eine Mail gesendet werden wenn Updates
von Passwoertern ausstehen (yes|no)

# Logging
MYNAME=$(basename $0);                                  # Dateiname des Scripts (nicht editieren)
LOGFILEPATH="/root/scripts/chage/log/";                 # Pfad zum skripteigenen Logfile
LOGFILENAME=$(hostname -f)"-checkpasswordage.log";      # Dateiname des skripteigenen Logfile
LOGFILE=${LOGFILEPATH}${LOGFILENAME};                   # Nicht editieren
LOGVERBOSITY="3";                                       # Standard-Loglevel [0 - 4]
SILENT_LVL="0";                                         # Definiert Silent (nicht editieren)
ERR_LVL="1";                                            # Definiert Loglevel von Error (nicht editieren)
WRN_LVL="2";                                            # Definiert Loglevel von Warn (nicht editieren)
INF_LVL="3";                                            # Definiert Loglevel von Info (nicht editieren)
DBG_LVL="4";                                            # Definiert Loglevel von Debug (nicht editieren)
```

```
## ------------------------------------------------------------------------------------------------------------
--
## Strings
## ------------------------------------------------------------------------------------------------------------
--

WORKLIST="";                                            # Nicht editieren
UPDATELIST="";                                          # Nicht editieren
ERRORSTATUSSTRING="";                                   # Nicht editieren
MYHOSTNAME=$(hostname -f);                              # Der Hostname des Systems auf dem das Skript
laeuft (nicht editieren)
SUBJECTSTATENOOP="[NOOP]";                              # Mail-Subject-Indikator fuer einen
erfolgreichen Lauf ohne Handlungsbedarf.
SUBJECTSTATEERROR="[ERROR]";                            # Mail-Subject-Indikator fuer einen Lauf bei dem
Fehler aufgetreten sind.
SUBJECTSTATEUPDATE="[PENDING-PASSWORD-UPDATE]";         # Mail-Subject-Indikator fuer Useraccounts die
einen Passwortwechsel benötigen.
MAILBODY="";


## ------------------------------------------------------------------------------------------------------------
--
## Funktionen
## ------------------------------------------------------------------------------------------------------------
--

# Ein formatiertes Datum bauen
function makeadate ()
{
        local l_FORMATTEDDATE="";
        l_FORMATTEDDATE=$(date +"%Y-%m-%d %H:%M:%S");
        echo "$l_FORMATTEDDATE";
}

# Die Userliste aus einer Datei auslesen und den Inhalt in ein Array befuellen, womit dann die WORKLIST
befuellt wird
function readUserList ()
{
        local l_FULLPATHTOUSERLIST="";
        l_FULLPATHTOUSERLIST="${PATHTOUSERLIST}${USERLISTFILE}";
        USERLIST=$(awk '{print $1}' $l_FULLPATHTOUSERLIST);
}


# Minimales Logging und Error-Handling
function notify () { log $SILENT_LVL "NOTE: $1"; }      # Wird immer ausgegeben
function error ()
{
        log $ERR_LVL "ERROR: $1";
}
function warn () { log $WRN_LVL "WARNING: $1"; }
function inf () { log $INF_LVL "INFO: $1"; }            # "info" ist ein Shell-Kommando
function debug () { log $DBG_LVL "DEBUG: $1"; }
function log ()
{
        local l_LOGDATE="";
        l_LOGDATE=$(makeadate);
        if [ $LOGVERBOSITY -ge $1 ]
        then
                # Ins Logfile schreiben
                echo -e "${l_LOGDATE} - $2" >> $LOGFILE;

                # An das Syslog uebergeben
                logger -t "${MYNAME}" "${2}";

                #Den Error-String erweitern
                if [ $1 = $ERR_LVL ]
                then
                        ERRORSTATUSSTRING+="  $2\n";
                fi
```

```
        fi

        # Die Meldung an STDOUT ausgeben
        echo "$2";
}

# Funktion um zu testen ob es den User gibt.
# Existierende User werden der WORKLIST hinzugefuegt.
# Nicht existierende User oder mehrfach vorkommende User loesen einen Fehler aus.
function initialtest ()
{
        # debug "initialtest (): Teste User: ${1}";

        local l_TESTCOUNT=$(grep -c "^${1}:" /etc/shadow);

        # debug "initialtest (): TESTCOUNT: ${l_TESTCOUNT}";

        if [ ${l_TESTCOUNT} -ne 1 ]
        then
                if [ ${l_TESTCOUNT} = 0 ]
                then
                        error "initialtest (): User ${1} existiert nicht.";
                else
                        error "initialtest (): User ${1} kommt ${l_TESTCOUNT} mal vor.";
                fi
        else
                # debug "initialtest (): User ${1} gefunden und ${l_TESTCOUNT} mal vorhanden.";
                inf "initialtest (): User ${1} existiert.";
                inf "Fuege User ${1} der WORKLIST hinzu.";
                WORKLIST+=" "${i};
        fi
}


# Funktion um das Alter der Userpasswoerter zu ueberpruefen
function checkPasswordAge ()
{
        # debug "checkPasswordAge (): Ermittle Passwortalter von User: ${1}";

        # init
        local l_ITEMARRAY="";
        local l_PARSESTRING="";
        local l_COUNTERVAR=0;
        local l_ITEMARRAY="";
        local l_LASTCHANGEOFPASSWORD="";
        local l_TODAY="";
        local l_DIFFERENCE="";

        # Den Eintrag fuer den uebergebenen USER aus der /etc/shadow auslesen
        l_PARSESTRING=$(grep "^$1:" /etc/shadow);
        # debug "checkPasswordAge (): ermittelter String: ${l_PARSESTRING}";

        # Den String von Asterisks saeubern
        l_PARSESTRING=$(echo "${l_PARSESTRING}" | sed 's/\*/ASTERISK/g');
        # debug "l_PARSESTRING nach SED: ${l_PARSESTRING}";

        # Die Textzeile aus /etc/shadow fuer das Array aufsplitten
        l_ARRAYPREPARE=$(echo "${l_PARSESTRING}" | tr ":" "\n");
        # debug "l_ARRAYPREPARE: ${l_ARRAYPREPARE}";

        # Das Array befuellen
                # 0 = Username
                # 1 = Passwort-hash
                # 2 = Alter des Passworts in Tagen seit 01.01.1970
        l_ITEMARRAY=($l_ARRAYPREPARE);
        # debug "[0]=${l_ITEMARRAY[0]}";
        # debug "[1]=${l_ITEMARRAY[1]}";
        # debug "[2]=${l_ITEMARRAY[2]}";

        # Timestamp der letzten Passwortaenderung
```

```
        l_LASTCHANGEOFPASSWORD=${l_ITEMARRAY[2]};
        # debug "checkPasswordAge (): Timestamp der letzen Passwortaenderung:
${l_LASTCHANGEOFPASSWORD}.";

        # Timestamp fuer heute erzeugen
        l_TODAY=$(($(date --utc --date "" +%s)/86400));
        # debug "checkPasswordAge (): Timestamp von heute: ${l_TODAY}.";

        # Berechnen der Differenz zwischen den Timestamps der letzten Passwortaenderung und heute
        l_DIFFERENCE=$((l_TODAY-l_LASTCHANGEOFPASSWORD));
        # inf "checkPasswordAge (): Die Differenz zwischen den Timestamps der letzen Passwortaenderung
und heute betraegt ${l_DIFFERENCE} Tage.";

        echo "${l_DIFFERENCE}";
}

function sendAlertMail ()
{
# Test ob eine Mail gesendet werden soll
if [ "${SENDMAILIFNOOP}" == "yes" ] || [ "${SENDMAILIFERROR}" == "yes" ] ||  [ "${SENDMAILIFUPDATE}" ==
"yes" ]
then
        notify "Mailen des Status vom ${SCRIPTSTARTDATE} gestartet.";
        inf "Konfiguration:";
        inf " * Mail bei NOOP: ${SENDMAILIFNOOP}";
        inf " * Mail bei Fehler: ${SENDMAILIFERROR}";
        inf " * Mail bei Updates: ${SENDMAILIFUPDATE}";

        local l_SUBJECTDATE=$SCRIPTSTARTDATE;
        local l_MAILSUBJECT="";
        local l_MAILBODY="";


        # Den Anfang des Betreff mit Hostname und Datum generieren.
        l_MAILSUBJECT="[${MYHOSTNAME}]: ${l_SUBJECTDATE} - ";

        # Den Status des Programmlaufs an den Betreff anhaengen.
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                # Wenn es keinen Fehler gab
                if [ "$UPDATELIST" = "" ]
                then
                        # Wenn keine Updates durchzufuehren sind (NOOP)
                        debug "sendAlertMail (): Es sind keine Updates auszufuehren, Subject um NOOP
erweitert.";
                        l_MAILSUBJECT+="${SUBJECTSTATENOOP}";
                else
                        # Es liegen Userpasswoerter vor, die geupdated werden muessen (UPDATE)
                        debug "sendAlertMail (): Es liegen Updates vor, Subject um Update erweitert.";
                        l_MAILSUBJECT+="${SUBJECTSTATEUPDATE}"
                fi

        else
                # Es sind Fehler aufgetreten (ERROR)
                debug "sendAlertmail (): Es liegen Fehler vor, Subject um ERROR erweitert.";
                l_MAILSUBJECT+="${SUBJECTSTATEERROR}"
        fi

        # Bauen des Mail-Bodies abhaengig vom Status
        if [ "$ERRORSTATUSSTRING" = "" ]
        then
                # Es sind keine Fehler aufgetreten
                if [ "$UPDATELIST" =  "" ]
                then
                        # Wenn keine Updates durchzufuehren sind (NOOP)
                        debug "sendAlertMail (): Es sind keine Updates auszufuehren, NOOP-Body
generieren.";
                        l_MAILBODY+="Beim Lauf des Skripts ${MYNAME} auf ${MYHOSTNAME} wurde keine
Operation ausgefuehrt.\n";
                        l_MAILBODY+="\n";
```

```
l_MAILBODY+="================================================================================================\
n"
                        l_MAILBODY+="Das Alter des Passworts aller ueberprueften User ist unterhalb des
Ablauflimits\n";
                        l_MAILBODY+="von ${MAXPASSWORDAGE} Tagen.\n";
l_MAILBODY+="================================================================================================\
n"
                else
                        # Es liegen Userpasswoerter vor, die geupdated werden muessen (UPDATE)
                        debug "sendAlertMail (): Es liegen Updates vor, UPDATE-Body generieren.";
                        l_MAILBODY+="Beim Lauf des Skripts ${MYNAME} auf ${MYHOSTNAME} wurden folgende
User gefunden,\n";
                        l_MAILBODY+="deren Passwort abgelaufen ist.\n";
                        l_MAILBODY+="\n";
l_MAILBODY+="================================================================================================\
n";
                        for i in $UPDATELIST; do
                                l_MAILBODY+="${i}\n";
                        done;
l_MAILBODY+="================================================================================================\
n";
                fi

        else
                # Es sind Fehler aufgetreten (ERROR)
                debug "sendAlertMail (): Es sind Fehler aufgetreten, ERROR-Body generieren.";
                l_MAILBODY+="Beim Lauf des Skripts ${MYNAME} auf ${MYHOSTNAME} traten folgende Fehler
auf:\n";
                l_MAILBODY+="\n";
l_MAILBODY+="================================================================================================\
n";
                l_MAILBODY+="${ERRORSTATUSSTRING}";
l_MAILBODY+="================================================================================================\
n";
        fi


        # Senden der E-Mail

        echo -e "${l_MAILBODY}" | mailx -s "${l_MAILSUBJECT}" -r $MAILSENDER $MAILRECEIVER && inf "Die
Email an ${MAILRECEIVER} wurde versendet." || error "Die Email an ${MAILRECEIVER} konnte nicht versendet
werden."

        notify "Mailen des Status der Passwortalterueberpruefung vom ${SCRIPTSTARTDATE} beendet."

fi
}

## ------------------------------------------------------------------------------------------------------
--
## Main
## ------------------------------------------------------------------------------------------------------
--

# Startup Info
SCRIPTSTARTDATE=$(makeadate);
notify
"************************************************************************************************";
notify "${SCRIPTSTARTDATE} - ${MYHOSTNAME}: ${MYNAME} gestartet."
notify
"************************************************************************************************";

# Die Userliste in USERLIST einlesen
readUserList;

# Initialer Test ob mehr als eine Trefferzeile von grep fuer einen User zurueckgeliefert wurde oder ob
ein
# User gar nicht existiert.
# Wenn ein User existiert, wird er der WORKLIST hinzugefuegt.
inf "Initiale Tests und Befuellung der WORKLIST.";
```

```
for i in $USERLIST; do
        initialtest $i;
done;

# Wenn keine Fehlerkondition vorliegt wird das Alter des Passworts, der in der WORKLIST gelisteten User,
ueberprueft. Wenn das Alter ueber der erlaubten
# Grenze ($MAXPASSWORDAGE) liegt, wird der User der UPDATELIST hinzugefuegt.
if [ "${ERRORSTATUSSTRING}" == "" ]
then
        inf "Die WORKLIST ist: ${WORKLIST}";

        # Aufruf der Funktion um das Alter der Passwoerter der einzelnen User aus USERLIST zu pruefen.
        inf "Abarbeitung der WORKLIST und ermitteln des Passwortalters.";
        for a in $WORKLIST; do
                USERPASSWORDAGE=$(checkPasswordAge $a);
                if [ "$USERPASSWORDAGE" -gt $MAXPASSWORDAGE ]
                then
                        # Wenn das Passwort abgelaufen ist
                        warn "Das Passwort von User $a ist $USERPASSWORDAGE Tage alt und ueber dem
maximalen Passwortalter von $MAXPASSWORDAGE Tagen.";
                        UPDATELIST+=" "${a};
                else
                        # Wenn das Passwort nicht abgelaufen ist
                        inf "Das Passwort von User $a ist $USERPASSWORDAGE Tage alt und unter dem
maximalen Passwortalter von $MAXPASSWORDAGE Tagen.";
                fi
        done;
fi

# Versenden der Mail
sendAlertMail;

# Skriptende Info
notify
"#############################################################################################################";
notify "$(makeadate) - ${MYHOSTNAME}: ${MYNAME} beendet."
notify
"#############################################################################################################\n\n"
;
```