Burp backup report generator script

This is a bash script that can be installed and run on a burp server in order to send out a daily (or so) backup report informing the backup admin if all backups have been updated in time or if some of them aren't updated anymore. read the description in the begining of the script for more details on the installation and configuration:

this script requires **GNU AWK** which under Ubuntu can be installed as package gawk. If you are using other implementations of awk such as mawk there may be problems with the regular expressions used in this script, as mawk does not understand the `\s` special character which stands for “zero to many spaces of any type”. If you can't install GNU Awk, you may want to replace the `\s` in the awk regex with `[ \t]` and use double quotes instead of single quotes around the argument, so for example `awk -F= “/^[\t ]*clientconfdir[ \t]*=/{print \$2}” “$serverConf”` which should do the trick.

WARNING This report generator currently completely ignores any warnings during the backups. So your backups could be incomplete with warnings (such as file vanished, or could not access file xy, in which case the file is not part of the backup) and this script won't notice that, as burp still considers those backups successful. I will have to spend some time in the future to include using `burp -a S` output or similar to find warnings in backups and add them to the reports.

burp-report.sh

```
#!/bin/bash
# based on the idea and some code of Thomas Constans' script found
here: https://infra.opendoor.fr/git/tom/BurpCheckAge
#
# this script creates a report for the admin showing the age of the
latest backup for each client
# The report is printed to STDOUT and the script will exit with code 0
if all backups are within limits
# or it will exit with exitcode equal to the numer of backups that are
over their age limit, so if
# 5 backups are too old, the exit code will be 5
#
# the default threshold time can be defined in the burp-server.conf
file which is located here:
serverConf="/etc/burp/burp-server.conf"
#
# this file should contain a parameter "warnadmin=<number of days>".
# if the client's config in clientconfdir contains a prameter
"warnadmin=<number of days>" it will overwrite
# the global defalt set in the server config.
# the backup will be considered too old if the age is more than <number
of days> old.
#
# "warnadmin" can also be set to "never" (or below zero) in which case
the age of the oldest backup of this
# client will only be included in the report for informational
```

```
purposes.
#
# run this script in a daily cronjob and pipe the output to a mail
sender of your choice, use the exit code to
# decide what the subject of your email should be. for example:
#
# report=$(/opt/burp-report.sh); ret=$?; echo "$report" | /usr/bin/mail
-s "$( [ $ret -eq 0 ] && echo "SUCCESS: all backups are within Limits"
|| echo "FAIL: $ret backups are too old")" backup@admin.local
#

[ -f $serverConf ] || { echo "could not find the burp server config in
$serverConf, please adjust the serverConf variable in this script
accordingly"; exit 1; }
clientconfdir=$(awk -F= '/^\s*clientconfdir\s*=/{print $2}'
"$serverConf")
if [ -z "$clientconfdir" ]; then
    echo "could not find a clientconfdir = parameter in $serverConf,
please define it before continuing with tihs script"
    exit 1
fi
recipients=$(awk -F= '/^\s*warn_recipients\s*=/{print $2}'
"$serverConf")
defaultWarnAge=$(awk -F= '/^\s*warnadmin\s*=/{print $2}' "$serverConf")
if [ -z "$defaultWarnAge" ]; then
    echo "Set a default maximum allowed age for your backups in
$serverConf, by setting the \"warnadmin = <number of days>\"
parameter."
    exit 1
fi

backupDir=$(awk -F= '/^\s*directory\s*=/{print $2}' "$serverConf")
if [ -z "$backupDir" ]; then
    echo "could not find the default backup directory in $serverConf,
make sure the \"directory\" parameter is set."
    exit 1
fi

okay=""
failed=""
info=""
maxClientNameLength=$( ls $clientconfdir/ | wc -L )
paddstring=$(printf '.%.0s' $(seq 1 $maxClientNameLength))"....."

for file in $clientconfdir/* ; do
    if [ ! -f $file ] ; then continue ; fi
    if ( ! grep -q warnadmin $file ) ; then
        maxage=defaultWarnAge
    else
        maxage=$(awk -F= '/^\s*warnadmin\s*=/{print $2}' $file | tr -d
```

```
" ")
        if [ "$maxage" == "never" ]; then
            maxage=-1
        fi
    fi
    userdir=$(awk -F= '/^\s*directory\s*=/{print $2}' $file)
    if [ -z "$userdir" ]; then
        userdir=$backupDir
    fi
    userdir=${userdir}/$(basename $file)
    timestampfile=${userdir}/current/timestamp
    let "maxage=$maxage*86400"
    #test -r $timestampfile || { mail_error $mail $not_found ; continue
; }
    test -r $timestampfile || { echo $mail $not_found ; continue ; }

    fileage=$(date -d "$(cut -d " " -f 2- $timestampfile)" +%s)

    now=$(date "+%s")
    let "delta=$now-$fileage"
    client=$(basename $file)
    line="$client${paddstring:${#client}}"$(printf '% 4d'
$(($delta/86400)))
    if [ $maxage -lt 0 ]; then
        info="${info}$line\n"
    elif [ $delta -gt $maxage ] ; then
        failed="$failed$line (Limit: $(($maxage/86400)))\n"
    else
        okay="$okay$line\n"
    fi
done

if [ -n "$failed" ]; then
    echo "Benchmarks OVER the critical Age Limit"
    echo "======================================"
    echo -e "$failed"
fi
if [ -n "$okay" ]; then
    echo "Benchmarks WITHIN critical Age Limit"
    echo "===================================="
    echo -e "$okay"
fi
if [ -n "$info" ]; then
    echo "Benchmarks WITHOUT critical Age Limit"
    echo "====================================="
    echo -e "$info"
fi

if [ -n "$failed" ]; then
    exit $(($(echo -e "$failed" | wc -l )-1))
else
```

```
    exit 0
fi
```