Enhanced Samba Logging

Samba logging appears to show you everything but what you want to know. There is a solution however, in the vfs full audit module.

One client wanted to monitor activity by individuals at their location, watching for unusual activity. They agreed that simply comparing activity on a weekly basis would be sufficient.

Add in /etc/samba/smb.conf. This can be global, or on a particular share

smb.conf.additional

```
# use the vfs full audit module
# see
https://moiristo.wordpress.com/2009/08/10/samba-logging-user-activity/
# https://www.samba.org/samba/docs/man/manpages-3/vfs_full_audit.8.html
vfs objects = full_audit

# Username | machine name | name of service
vfs_full_audit:prefix = %U|%m|%S

## log the following functions
## create a directory, rename something, delete (unlink) a file, rm a
directroy
## read/write a file, open a file
full_audit:success = mkdir rename unlink rmdir pwrite open rmdir pread
# rename open !strict_unlock !pread !get_alloc_size !readdir !telldir
!lstat !closedir !connectpath !opendir

## Do not log anything on failure
full_audit:failure = none

## use syslog local7 for the logs
## you must create this in syslog.conf by adding a line
## local7.* /var/log/samba/log.audit
## and also set logrotate
full_audit:facility = local7

## all set to NOTICE
full_audit:priority = NOTICE
```

append to /etc/rsyslog.conf

rsyslog.conf.append

```
# set to log samba vfs audit module


```

```
local7.* /var/log/samba/audit.log
```

Append to /etc/logrotate.d/samba

samba.append

```
/var/log/samba/audit.log {
        daily
        missingok
        rotate 7
        postrotate
        invoke-rc.d rsyslog rotate > /dev/null
                /etc/init.d/samba reload > /dev/null
        endscript
        compress
        notifempty
}
```

```
/etc/init.d/rsyslog restart
/etc/init.d/samba restart
```

The following script (which is really too large for a KB article) can be run to summarize the log file. I put it in /opt/scripts, and store the results in /opt/smbaudit/. I then run the script via cron at 8am every morning (on audit.log.1, which is the previous days work after logrotate does its thing).

Two things about the code: 1. I have never gotten the hang of the correct syntax for going through a hash of hashes 2. I wrote it in a couple of hours, so it is definitely **not** the best code I have ever written

processSMBaudit.pl

```
#! /usr/bin/env perl

# copyright 2017, R. W. Rodolico
# use it as you want, make money from it, sell it for all I care,
Rename it, remove my name, whatever
#
# You could write the same
# thing in a few hours so this is true open source; I don't want
anything except to have the right to
# use it myself.
# BTW, I have no responsibility if it destroys your server, your
marriage, or anything else. Use at your own
# risk.

# script to process samba audit log, storing summary of information
# for future processing.
# will read $summaryFile into memory, if it exists, then scan
```

```
# $auditFile, adding new entries.
# will then write results back to $summaryFile (making a backup first)
# resulting file is a tab delimited text file, where each line begins
# with three standard fields; username, timestamp and IP.
# the remaining fields are actions based on %headers below (other
actions
# are ignored).
#
# NOTE: timestamp is floor'd to the nearest day, ie
int(timestamp/86400)*86400
# as we want to summarize a days activity.

use strict;
use warnings;
use Parse::Syslog;
# apt-get install libparse-syslog-perl

# path to Samba audit file
my $auditFile = '/var/log/samba/audit.log';
# path to our historical summary file
my $summaryFile = '/opt/smbaudit/samba_audit.summary';
# hash to store all our activity
my %activity;
# hash containing the action headers we care about
my %headers = ( 'mkdir' => 1, 'pread' => 1, 'pwrite' => 1,'rmdir' =>
1,'unlink' => 1 );
# number of seconds in a day, 86400. BREAKS on Daylight Savings Time
my $secondsInDay = 60*60*24;

# function loads the summary file into %activity
# also modified %headers, based on the headers it finds in the summary
file
sub loadSummary {
   open SUMMARY,"<$summaryFile" or return;
   print STDERR "Loading summary file\n";
   # file exists, so read in the first line, which is the headers
   my $line = <SUMMARY>;
   chomp $line;
   my @headers = split( "\t", $line );
   # replace our preset ones with whatever headers we have here
   %headers = map{ $_ => 1 } @headers;
   # the following are not actions, so remove them from the headers
   delete @headers{qw(user day ip)};
   # read each line and create the activity.
   # note that user, timestamp and IP are required to be in the first
   # three columns
   while ( $line = <SUMMARY> ) {
      chomp $line;
      my @data = split( "\t", $line );
      for ( my $i = 3; $i < @headers; $i++ ) {
         $activity{$data[0]}{$data[1]}{$data[2]}{$headers[$i]} =
```

```
$data[$i];
      }
   }
   close SUMMARY;
}

# get our summary file into the access hash
&loadSummary();

# use Parse::Syslog to read in each line, mainly to get the timestamp
my $parser = Parse::Syslog->new( $auditFile );
while ( my $sl = $parser->next ) {
   # remove the time from it; just date
   my $timestamp = int( $sl->{'timestamp'} / $secondsInDay ) *
$secondsInDay;
   # text contains the information we care about
   my @temp = split( '\|', $sl->{'text'} );
   # and we only care about the first three of them, which are user, ip
   # and the action they took
   my ( $user, $ip, $action ) = @temp[0..2];
   # update %activity if this is an action we track
   $activity{$user}{$timestamp}{$ip}{$action}++ if $headers{$action};
}

# make a backup of the summary file, if it exists
unlink "$summaryFile~" if -e "$summaryFile~";
rename $summaryFile, "$summaryFile~" if -e $summaryFile;
# and overwrite it
open SUMMARY, ">$summaryFile" or die "Could not save summary file
$summaryFile: $!\n";
my @line; # the line we'll build for output
# header line
print SUMMARY "user\tday\tip\t" . join( "\t", sort keys %headers ) .
"\n";
# process each user
foreach my $user ( sort keys %activity ) {
   push @line, $user;
   my $timestamp = $activity{$user};
   # process the date
   foreach my $time ( sort keys %$timestamp ) {
      push @line, $time;
      my $ips = $$timestamp{$time};
      # and the time
      foreach my $ip ( sort keys %$ips ) {
         push @line, $ip;
         my $actions = $$ips{$ip};
         # Finally, get the actions (all of them)
         foreach my $action ( sort keys %headers ) {
            push @line, $$actions{$action} ? $$actions{$action} : 0;
         }
         # finished with all the actions, so dump the line
```

```
            print SUMMARY join( "\t", @line ) . "\n";
            # and delete all the actions for a new IP
            delete @line[2..$#line];
         } # ip
         # delete the IP also for a new timestamp
         delete @line[1..$#line];
      } # timestamp
      # completely reset @line for a new user
      @line = ();
} # user
close SUMMARY;

1;
```