Synchronizing Users

For now, we're not going to go into LDAP or anything. Assuming you have a small shop with a small number of machines and a small number of users, and you just want things to be similar across the systems.

Standardizing Users

Ok, this part is not for synchronizing users, but actually for creating a list of standard users. The following Perl script is very insecure as it stores passwords and public keys, but at least they are encrypted.

The two variables at the top of the page, *%fixUserNames* and %passwords contain all of the information to be set up on the server. *%fixUserNames* either removes users or, if a new name is given, renames a user. *%passwords* creates a new user if they don't exist, adds their public ssh key and sets the password (changing the password if it is already set).

It does **not** set the UID, and sets primary group to *users*. All users are members of the group sudo, which gives them sudo rights. See line in middle of sub addAUser to modify that.

The passwords are encrypted using the command

```
echo 'mypassword' | openssl passwd -1 -stdin
```

where 'mypassword' is the password you want to give the user.

Use with caution, but it has worked well for us in the past. It is not well documented.

fixusers.pl

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

use strict;
use warnings;
use Data::Dumper;

# set this to 1 to not really do anything, but to only print what we
# would have done.
my $TEST = 0;


# we use this to know to change usernames, in other words, if
# the username user exists on the server, we need to change it
# to user1, while baduser is removed if it exists.
# if the $changeTo is empty, we simply remove the user.
my %fixUserNames = (
   'user'       => 'user1',
```

```
   'you'       => 'me',
   'baduser'    => '',
   );

# 'password' comes from the command
# echo 'mypassword' | openssl passwd -1 -stdin
# 'ssh key' is the value found in ~/.ssh/id_rsa.pub
# in the following example, 'user1' is created if they don't exist,
# and their password is changed to 'mypassword' (encrypted version
given).
# nothing is done to their ssh key.
# user2 is treated the same way, but their ssh public key is added to
their authorized_keys file
my %passwords = (
      'user1'  => {
                           'password' =>
'$1$hpr.bGjU$VgEWjkSIWZS.jlgxDRnCd0',
             'ssh key'  =>   ''
                        },
      'user2'    => {
            'password' =>   '$1$hpr.bGjU$VgEWjkSIWZS.jlgxDRnCd0',
            'ssh key'  =>    'ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAACAQDQ9nff4EfBKDe7R+/oyCjjNMsQ8FFBIKfc4wBr52i
0DXUVPiaj0eSRSBlwR5b6uYbHUUSGgC5hdn/L5iwe1QbKbP0I4h4EZYrSnzIWGkYPBk61hW
p2Oi3tOP8omQMh1e6u8OJhAeZ0EGy242BUABZ0rVyrNXpj1I3vaVhtnXLmUgTOzUxHJtQFt
6/PBwOGdP7K4EchNefAd+Ci7PYDxdCAaRYJE7u27Ua4//0xOfPBJ4YF4jTk39+9zpbe7foL
qDRwddPcZX7uzZ+BpguQVhsdR3H18owxuTFoy+NZEfQ4JJPwWyHspbBZ9gW82oK3WLptnJM
Pd5W8ZkJ89k8bwarKBLgH26OwrEXx8EwdrH00CY0HOErClzO8RJ76ViaqpkwI1VTbUYMKiC
pJcl0xFvP0j4H3WGrptBgIOMAUYmfXP/0AcohVa8SJ+SX9NxFFqsoZynjy5FFUzombDSP7L
pU9euBTgfc0ihlL31O9mHVmvZCeG0EzzcN++do8YxxzdpKZhYSJNGNgpFuauyZwm62emout
0u14T1PZi5iqqdF9p9LZudcLEnneY7ofnfyK92K5HBckhWyUS2ztLuwRab0FrmuOlCvyKV7
VSXxgocdJNnKEln4A0FwnUG7awJShbD6332MnKeJM1D6dnEkQo2WSOpi944wK9UAe72wZCR
aq3Q== user1@example'
                        },
   );


my $tempFile = '/tmp/tempPassword';

sub runCommand {
   my $command = shift;
   if ( $TEST == 1 ) {
      print "$command\n";
      return;
   } else {
      system($command);
      return $? >> 8;
   }
}

sub getUserHomeDir {
```

```
    my $user = shift;
    return (getpwnam($user))[7];
}


# checks to see if username is on this system
# returns true if username exists, false if not
sub userExists {
    my $username = shift; # get whatever username they pass in
    system( "id -u $username > /dev/null 2>&1" );
    return !( $? >> 8 );
}

# create a user on the system
# if $makeHomeDir is true, will create home directory
sub addAUser {
    my $username = shift; # the username we want to add
    my $makeHomeDir = shift; # choose whether to make homedir
    # makes root of homedir if not there
    &runCommand( 'mkdir -p /home/users' ) unless -e '/home/users';
    # build our command
    my $command = "useradd --no-user-group --gid users --base-dir
/home/users --groups sudo --shell /bin/bash";
    # add flag to not create homedir if they told us to
    if ( $makeHomeDir ) {
        $command .= ' --create-home';
    } else {
        $command .= ' --no-create-home' unless $makeHomeDir;
    }
    # add the username
    $command .= " $username";
    # execute the command
    &runCommand($command);
     # check if it works
    my $success = &userExists( $username );
    # see if worked
    return $success;
}

# rename a user
sub renameUsers {
    my $lookingFor = shift; # this is the account we want to rename
    my $changeTo = shift; # this is what we want to rename it to
    # does the account to rename exist?
    if ( &userExists( $lookingFor ) ) {
        print "Changing $lookingFor\n";
        # does the account we want to rename to not exist?
        if ( not userExists( $changeTo ) ) {
            # then, add the new user account
            if ( $changeTo ) {
                if ( &addAUser( $changeTo, 0 ) ) {
```

```
                # mv the old home directory to the new home directory
name
                &runCommand("mv ~$lookingFor ~$changeTo");
                # change the ownership of all files to new user
                &runCommand("chown -fR $changeTo:users ~$changeTo");
             } else { # we failed to add the user!
                return 0; # so, return false
             } # if..else
          } # if changeTo exists
          print "\tDeleting $lookingFor\n";
       } # if
          # we succeeded, so delete the original user account
          &runCommand("deluser --remove-home $lookingFor");
          &runCommand("delgroup --only-if-empty $lookingFor" );
   } # if $lookingFor exist
   return 1; # yay, we succeeded, so return true
}

sub setUpSSH {
   my $user = shift;
   my $passphrase = shift;

   return 3 unless defined( $passphrase ) && $passphrase;
   my $dir = &getUserHomeDir( $user );
   return 0 unless defined( $dir ) && $dir && -d "$dir";
   return 1 if -e "$dir/.ssh/authorized_keys";
   &runCommand( "mkdir -p $dir/.ssh" );
   &runCommand( "chmod 700 $dir/.ssh" );
   open KEY,">$dir/.ssh/authorized_keys" or die "could not create
$dir/.ssh/authorized_keys: $!\n";
   print KEY $passphrase;
   close KEY;
   &runCommand( "chmod 600 $dir/.ssh/authorized_keys" );
   &runCommand( "chown $user:users -fR $dir/.ssh" );
   return 2;
}


# check for and rename users in the fixUserNames hash
# removes any users who have an empty username
foreach my $user (keys %fixUserNames ) {
   my $worked = &renameUsers( $user, $fixUserNames{$user} );
   if ( not $worked ) {
      warn "Could not rename $user to $fixUserNames{$user}\n";
   }
}

# add any accounts which do not exist. Also, build a password file
# for setting passwords
open PASS, ">$tempFile" or die "could not create $tempFile: $!\n";
foreach my $user ( keys %passwords ) {
```

```
   unless ( &userExists( $user ) ) {
     warn "Could not add user $user\n" unless &addAUser( $user, 1 );
   }
   my $ssh = &setUpSSH( $user, $passwords{$user}{'ssh key'} );
   warn "User $user already has an authorized_keys file\n" if $ssh ==
1;
   warn "User $user does not have a home directory\n" if $ssh == 0;
   warn "User $user has no ssh key set in config\n" if $ssh == 4;
   # print "User $user set up for ssh public key\n" if $ssh == 2;
   if ( &userExists( $user ) ) { # the user has a home directory
      print PASS join( ':', ($user,$passwords{$user}{'password'}) ) .
"\n";
   }
}
close PASS;

# change all passwords in $tempFile
&runCommand( "chpasswd -e < $tempFile" );
#unlink $tempFile;

1;
```