Contents

1 Description of the problem 2
  1.1 Insecure .................................................. 2
  1.2 userPassword and samba password may be corrupted .... 2
  1.3 requires slappasswd .................................... 2

2 Solution 2
  2.1 enable taint mode ...................................... 2
  2.2 Model of changing passwords fixed ........................ 4
  2.3 generate hashes with perl libraries ..................... 5
    2.3.1 new perl function substitute for(slappasswd hash gen-
    erator ........................................................... 5
  2.4 required smbldap.conf modification .................... 7

3 Modification examples 7
  3.1 A full replacement for /bin/passwd .................... 7
  3.2 Script for users who want to update password in LDAP and
       local passwd file .......................................... 8

*freebsd port: net/smbldap-tools
†samba.idealx.org
‡wielebap@iem.pw.edu.pl
1 Description of the problem

This document can be downloaded from:


1.1 Insecure

Script *smbldap passwd* cannot be run with perl -T (taint) option turned on. Therefore it is not possible to modify this script and run it in setuid mode.

1.2 userPassword and samba password may be corrupted

If *slapasswd* is not available *userPassword* field is still modified with the empty password!!! and samba password is still modified with the entered password.

1.3 requires slapasswd

Script *smbldap passwd* requires *slapasswd* to generate password and this is not configurable.

2 Solution

This section describes what was done to make *smbldap passwd* to be secure.

2.1 enable taint mode

I’ve turned on perl taint mode:

< #!/usr/bin/perl -w
---
> #!/usr/bin/perl -wT

In perl taint mode it is not possible to use pragma:

use lib "@RealBin/";

Therefore FindBin package is not used now, and these pragmas where deleted:
The `use smbldap_tools;` pragma was replaced to import only required functions to be sure they are secure and do not change eg. `euid = 0`.

```
< use smbldap_tools;
---
  > use smbldap_tools qw(connect_ldap_master
  >       get_user_dn
  >       get_dn_from_line
  >       is_samba_user
  >       is_user_valid
  >       %config);

Generating password via `slappasswd` was insecure as the external programme had to be invoked. Now this `slappasswd` is run in a child process and shell is not used.

```
< my $hash_password = '';
< if ( ($config{hash_encrypt} eq "CRYPT" && defined($config{crypt_salt_format})) )
  {
  <   $hash_password = `/usr/local/sbin/slappasswd -h {$config{hash_encrypt}} -c '
  $config{crypt_salt_format}' -s '$pass'';
  < } else {
  <   $hash_password = `/usr/local/sbin/slappasswd -h {$config{hash_encrypt}} -s '
  $pass'';
  < } }
< chomp($hash_password);
---
  > # Prepare '$hash_password' for 'userPassword'
  > my $hash_password;
  > # Generate password hash
  > if (!$config{without_slappasswd}) {
  >   # checking if password is tainted: nothing is changed!!!!
  >   # essential for perl 5.8
  >   ($pass =~ /^(.*)$/ and $pass=$1) or
  >   die "$0: user password is tainted\n";
  >   # use slappasswd to generate hash
  >   if ( ($config{hash_encrypt} eq "CRYPT" && defined($config{crypt_salt_format})
  >        ) ) {

```
A new script also checks if the password hash was generated. It dies otherwise.

2.2 Model of changing passwords fixed

- Now userPassword hash is generated before samba password is updated.
- And userPassword field is updated after successful changing samba password.
- A new script checks if the password hash was generated properly. It dies otherwise.

``` perl
< chomp($hash_password);
---
> # check if a hash was generated, otherwise die
> defined($hash_password) or
> die "I cannot generate the proper hash!\n";
> chomp($hash_password);
```
2.3 generate hashes with perl libraries

A new, very useful option was added. It is possible not to use slappasswd tool to generate password hashes but only perl libraries.

Therefore two new subroutines were added:

> # function declaration
> sub make_hash;
> sub make_salt;

The `make_hash` subroutine can generate the same hashes as `slappasswd` utility does.

New modules were added as they are required by `make_hash`:

> use Digest::MD5 qw(md5);
> use Digest::SHA1 qw(sha1);
> use MIME::Base64 qw(encode_base64);

2.3.1 new perl function substitute for slappasswd hash generator

I have created a new function which completely depends on perl built-in functions and cpan modules.

Using of this function is much safer than `slappasswd` as no external programmes are invoked.

Now you can use smbldap-passwd script on the platform where there is no `slappasswd` script.

The code of a new make_hash function:

> # Generates hash to be one of the following RFC 2307 schemes:
> # CRYPT, MD5, SMD5, SHA, SSHA, and CLEARTEXT
> # SSHA is default
> # '%s' is a default crypt_salt_format
> # A substitute for slappasswd tool
> sub make_hash
> {
> my $hash_encrypt;
> my $crypt_salt_format;
> my $clear_pass=$_[0] or return undef;
> $hash_encrypt='{'. $[1] . '}' or $hash_encrypt = "{SSHA}";
> $crypt_salt_format=$_[2] or $crypt_salt_format = '%s';
> }
my $hash_pass;
if ($hash_encrypt eq "{CRYPT}" && defined($crypt_salt_format)) {
    # Generate CRYPT hash
    # for unix md5crypt $crypt_salt_format = ’$1$%.8s’
    my $salt = sprintf($crypt_salt_format,make_salt());
    $hash_pass = "{CRYPT}" . crypt($clear_pass,$salt);
}
elsesif ($hash_encrypt eq "{MD5}"") {
    # Generate MD5 hash
    $hash_pass = "{MD5}" . encode_base64( md5($clear_pass),'' );
}
elsesif ($hash_encrypt eq "{SMD5}"") {
    # Generate SMD5 hash (MD5 with salt)
    my $salt = make_salt(4);
    $hash_pass = "{SMD5}" .
        encode_base64( md5($clear_pass . $salt) . $salt,'');
}
elsesif ($hash_encrypt eq "{SHA}"") {
    # Generate SHA1 hash
    $hash_pass = "{SHA}" . encode_base64( sha1($clear_pass),'' );
}
elsesif ($hash_encrypt eq "{SSHA}"") {
    # Generate SSHA hash (SHA1 with salt)
    my $salt = make_salt(4);
    $hash_pass = "{SSHA}" .
        encode_base64( sha1($clear_pass . $salt) . $salt,'');
}
elsesif ($hash_encrypt eq "{CLEARTEXT}"") {
    $hash_pass=$clear_pass;
} else {
    $hash_pass=undef;
}
return $hash_pass;
}

# Generates salt
# Similar to Crypt::Salt module from CPAN
sub make_salt {
    my $length=32;
    $length = $_[0] if exists($_[0]);
    my @tab = ('.', '/', 0..9, 'A'..'Z', 'a'..'z');
>  return join "", @tab{map {rand 64} (1..$length)};
> }

2.4 required smbldap.conf modification

The patch for smbldap.conf

--- smbldap.conf.orig Mon Oct 11 15:33:20 2004
+++ smbldap.conf Mon Oct 11 15:34:25 2004
@@ -106,7 +106,7 @@
    scope="sub"

-    # Unix password encryption (CRYPT, MD5, SMD5, SSHA, SHA)
+    # Unix password encryption (CRYPT, MD5, SMD5, SSHA, SHA, CLEARTEXT)
    hash_encrypt="SSHA"

    # if hash_encrypt is set to CRYPT, you may set a salt format.
@@ -190,3 +190,9 @@
    # prefer Crypt::SmbHash library
        with_smbpasswd="0"
        /usr/bin/smbpasswd"
+    +# Allows not tu use slappasswd (if without_slappasswd == 1 in smbldap_conf.pm)
+    +# but prefer Crypt:: libraries
+        without_slappasswd="0"
+        /usr/local/sbin/slappasswd"
+        

3 Modification examples

3.1 A full replacement for /bin/passwd

Script checks if the user is in LDAP, if not it only invokes local passwd. If the user is in LDAP the script changes userPassword field and samba hashes.

This script can be downloaded from:

    www.iem.pw.edu.pl/~wielebap/ldap/smbldap-tools/newpasswd

Remember to apply smbldap.conf patch which was described above.
3.2 Script for users who want to update password in LDAP and local passwd file

This script is designed for users who are migrating from /etc/passwd to LDAP.

The script can be downloaded from:

www.iem.pw.edu.pl/~wielebap/ldap/smbldap-tools/newpasswd-migrate

This script is almost the same as mentioned in the previous subsection. The difference is included in the patch:

--- newpasswd Thu Oct 19 10:56:30 2004
+++ newpasswd-migrate Thu Oct 19 10:51:10 2004
@@ -244,6 +244,19 @@
     warn "Unable to change password : ", $modify->error ;
     # take down session
     $ldap_master->unbind;
    +} else {
    +  # take down session
    +  $ldap_master->unbind;
    +  # become root & change password in master.passwd
    +  >=0; # euid = 0
    +  open BUF,"|-
    +  exec "/usr/sbin/pw","usermod","$user","-h","0";
    +  local $SIG{PIPE} = sub {die "buffer pipe terminated" }
    +  print BUF "" . $pass . "\n"
    +  close BUF or die "master.passwd password has not been changed\n"
    +  =$<; # euid = uid
    +  print "Your password has been changed.\n"
    }

exit 0;

The patch can be downloaded from:

www.iem.pw.edu.pl/~wielebap/ldap/smbldap-tools/newpasswd.diff

Remember to apply smbldap.conf patch which was described in the previous section.