Rocky Enterprise Linux 9.2 Manual Pages on command 'authselect-profiles.5'

$ man authselect-profiles.5

AUTHSELECT-PROFILES(5) AUTHSELECT-PROFILES(5)

NAME

authselect-profiles - how to extend authselect profiles.

DESCRIPTION

This manual page explains how are authselect profiles organized and how to create new profiles.

PROFILE DIRECTORIES

Profiles can be found in one of three directories.

/usr/share/authselect/default

Read-only directory containing profiles shipped together with authselect.

/usr/share/authselect/vendor

Read-only directory for vendor-specific profiles that can override the ones in default directory.

/etc/authselect/custom

Place for administrator-defined profiles.

PROFILE FILES

Each profile consists of one or more of these files which provide a

mandatory profile description and describe the changes that are done to the system.

README

Description of the profile. The first line must be a name of the profile.

system-auth

PAM stack that is included from nearly all individual service configuration files.

password-auth, smartcard-auth, fingerprint-auth

These PAM stacks are for applications which handle authentication from different types of devices via simultaneously running individual conversations instead of one aggregate conversation.

postlogin

The purpose of this PAM stack is to provide a common place for all PAM modules which should be called after the stack configured in system-auth or the other common PAM configuration files. It is included from all individual service configuration files that provide login service with shell or file access. NOTE: the modules in the postlogin configuration file are executed regardless of the success or failure of the modules in the system-auth configuration file.

nsswitch.conf

Name Service Switch configuration file. Only maps relevant to the profile must be set. Maps that are not specified by the profile are included from /etc/authselect/user-nsswitch.conf.

dconf-db

Changes to dconf database. The main uses case of this file is to set changes for gnome login screen in order to enable or disable smartcard and fingerprint authentication.

dconf-locks

This file define locks on values set in dconf database.

CONDITIONAL LINES

Each of these files serves as a template. A template is a plain text

file with optional usage of several operators that can be used to provide some optional profile features.

{continue if "feature"}

Immediately stop processing of the file unless "feature" is defined (the rest of the file content will be removed). If "feature" is defined, the whole line with this operator will be removed and the rest of the template will be processed.

{stop if "feature"}

Opposite of "continue if". Immediately stop processing of the file if "feature" is defined (the rest of the file content will be removed). If "feature" is not defined, the whole line with this operator will be removed and the rest of the template will be processed.

{include if "feature"}

Include the line where this operator is placed only if "feature" is defined.

{exclude if "feature"}

Opposite to "include-if". Include the line where this operator is placed only if "feature" is not defined.

{imply "implied-feature" if "feature"}

Enable feature "implied-feature" if feature "feature" is enabled. The whole line with this operator is removed, thus it is not possible to add anything else around this operator at the same line.

{if "feature":true|false}

If "feature" is defined, replace this operator with string "true", otherwise with string "false".

{if "feature":true}

If "feature" is defined, replace this operator with string "true", otherwise with an empty string.

It is also possible to use logical expression in conditional line instead of specifying single feature name. In this case the expression will evaluate to true or false and the conditional operator will act

upon the result.

The expression syntax consists of feature names (e.g. "feature") which returns true if the feature is defined or false if it is not defined and from the following logical operators: and, or and not. The expression may also be enclosed in parentheses and contain multiple subexpressions.

For example:

{if "feature1" or "feature2":true}

If "feature1" or "feature2" is defined, replace this operator with string "true", otherwise with an empty string.

{if not "feature":true|false}

If "feature" is not defined, replace this operator with string "true", otherwise with string "false".

{if not "feature":true}

If "feature" is not defined, replace this operator with string "true", otherwise with an empty string.

{if "feature1" and ("feature2" or "feature3"):true}

If "feature1" is defined, and one of "feature2" and "feature3" is defined replace this operator with string "true", otherwise with an empty string.

EXAMPLE

Here is an example of using "if" operator. If "with-sudo" feature is enabled, it will add "sss" to sudoers line.

```
passwd:     sss files
group:      sss files
netgroup:   sss files
automount:  sss files
services:   sss files
sudoers:    files {if "with-sudo":sss}
```

Here is an example of "continue-if" and "include-if" operators. The resulting file will be empty unless "with-smartcard" feature is enabled. If it is enabled and also "with-faillock" feature is enabled, it will also enable support for pam_faillock.

```
        {continue if "with-smartcard"}
        auth        required                      pam_env.so
        auth        required                      pam_faildelay.so delay=2000000
          auth        required                            pam_faillock.so preauth silent deny=4 unlock_time=1200 {include if
"with-faillock"}
        auth        [default=1 ignore=ignore success=ok]        pam_succeed_if.so uid >= 1000 quiet
        auth        [default=1 ignore=ignore success=ok]        pam_localuser.so
        auth        sufficient                    pam_unix.so nullok
        auth        requisite                     pam_succeed_if.so uid >= 1000 quiet_success
        auth        sufficient                    pam_sss.so forward_pass
          auth        required                            pam_faillock.so authfail deny=4 unlock_time=1200        {include if
"with-faillock"}
        auth        required                      pam_deny.so
        ...
      Here is an example of "continue-if" using logical expression. The file
      will be empty unless "with-smartcard" or "with-smartcard-required" is
      set. This will simplify the call of authselect select command which
      does not have to include both features but only
      "with-smartcard-required" is necessary.
        {continue if "with-smartcard" or "with-smartcard-required"}
        auth        required                      pam_env.so
        auth        required                      pam_faildelay.so delay=2000000
          auth        required                            pam_faillock.so preauth silent deny=4 unlock_time=1200 {include if
"with-faillock"}
        auth        [default=1 ignore=ignore success=ok]        pam_succeed_if.so uid >= 1000 quiet
        auth        [default=1 ignore=ignore success=ok]        pam_localuser.so
        auth        sufficient                    pam_unix.so nullok
        auth        requisite                     pam_succeed_if.so uid >= 1000 quiet_success
        auth        sufficient                    pam_sss.so forward_pass
          auth        required                            pam_faillock.so authfail deny=4 unlock_time=1200        {include if
"with-faillock"}
        auth        required                      pam_deny.so
        ...
```

Here is an example of "imply-if" operator. Enabling feature "with-smartcard-required" will also enable "with-smartcard" to make sure that all relevant PAM modules are used. This will achieve the same behavior as the previous example.

```
{imply "with-smartcard" if "with-smartcard-required"}
auth        required                    pam_env.so
auth        required                    pam_faildelay.so delay=2000000
auth        [success=1 default=ignore]  pam_succeed_if.so service notin login:gdm:xdm:kdm:kde:xscreensaver:gnome-screensaver:kscreensaver quiet use_uid {include if "with-smartcard-required"}
auth        [success=done ignore=ignore default=die]  pam_sss.so require_cert_auth ignore_authinfo_unavail {include if "with-smartcard-required"}
auth        [default=1 ignore=ignore success=ok]  pam_succeed_if.so uid >= 1000 quiet
auth        [default=1 ignore=ignore success=ok]  pam_localuser.so  {exclude if "with-smartcard"}
auth        [default=2 ignore=ignore success=ok]  pam_localuser.so  {include if "with-smartcard"}
auth        [success=done authinfo_unavail=ignore user_unknown=ignore ignore=ignore default=die] pam_sss.so try_cert_auth {include if "with-smartcard"}
auth        sufficient                  pam_unix.so {if not "without-nullok":nullok}
auth        requisite                   pam_succeed_if.so uid >= 1000 quiet_success
auth        sufficient                  pam_sss.so forward_pass
auth        required                    pam_deny.so
...
```

CREATING A NEW PROFILE

To register a new profile within authselect, create a directory in one of the authselect profile locations with the files listed above. Not all of the files must be present, only README is mandatory. Other files can be created on per-need basis.

You may find authselect create-profile command helpful when creating new profile. See authselect(8) manual page or authselect create-profile --help for more information.

SEE ALSO

authselect(8), nsswitch.conf(5), PAM(8)