PAM OPENOTP PLUGIN

PAM OpenOTP Plugin

LDAP PAM NSS SSH

How To Install and Configure PAM OpenOTP Plugin to Enable Multifactor Authentication on Linux Machines

Simple login flow

Push Login flow

1. Background

On Unix-like systems, processes such as the OpenSSH daemon need to authenticate the user and learn a few things about him or her (user ID, home directory, …). Authentication is done through a mechanism called Pluggable Authentication Modules, and retrieving information about users (or even groups, hostnames, …) is done through another mechanism, called the Name Service Switch.

In this tutorial, we will allow users accounts stored as posixAccount objects in an LDAP server to be considered valid in a system, in addition to the locally-defined ones, by configuring NSS. We will then configure PAM to delegate authentication to OpenOTP accounts stored on the LDAP server.

When authenticating a user through PAM, a process will read /etc/pam.d/, where is the name of the service that the process implements. Each line in that file mentions a module to load, how to use it, and what decision to make based on the result. We will install the pam_openotp.so module and add a line mandating its use to contact your OpenOTP server. In CentOS 6, most service-specific files actually defer to one of two generic files, password-auth and system-auth, and we will modify them.

Processes that need to find out, say, what users or groups exist, use a set of functions implemented by the C standard library. These functions will read the file /etc/nsswitch.conf and, according to the information it contains, load modules in the form of libraries and gets the needed information through these modules in a certain order. We will install the NSS -based spankey module to collect user account information on our LDAP server. If pam_ldap module returns the account information you don't need to install spankey, this particular case is not explained in this documentation.

pam_ldap.so, nss_ldap.so and nslcd are not maintained by RCDevs. The first was written by PADL Software Pty Ltd: PAM LDAP, and the latter two are part of the nss-pam-ldapd package, maintained by Arthur de Jong NSS PAM LDAP.

2. Prerequisites

Firstly, you must have a configured OpenOTP and SpanKey server available through WebADM.

Then you have to install spankey_client, pam_openotp and & nscd packages on your server on which you want 2 Factor Authentication. All packages are available on RCDevs Repository. The best way is to configure RCDevs Repository to install every package with our repository.

On a RedHat, CentOS or Fedora system, you can use our repository, which simplifies updates. Add the repository:

```
yum install https://www.rcdevs.com/repos/redhat/rcdevs_release-1.0.0-0.noarch.rpm
```

You are now able to install RCDevs packages on your system:

```
yum install pam_openotp nscd spankey_client
```

On a Debian and Ubuntu system, you can use our repository, which simplifies updates. Add the repository:

```
wget https://www.rcdevs.com/repos/debian/rcdevs-release_1.0.1-0_all.deb
apt-get install ./rcdevs-release_1.0.1-0_all.deb
```

Update apt cache:

```
apt-get update
```

You are now able to install RCDevs packages on your system:

```
apt-get install pam-openotp nscd spankey-client
```

After downloading and installing the previous packages, we can start the configuration of these different products.

3. WebADM Accounts Configuration

To use your LDAP account on UNIX servers, you have to extend your account to UNIX through WebADM GUI. To extend your account to UNIX, click on your account on the left tree, you can find on the user details, the option `Add Extension`, select `UNIX Account` and click on `Add` button.

You will see the following screen after clicking `Add`:

> **Note**
>
> At this step, be careful to not use a UID already assigned to an existing user. We advise starting from uid=1000…

Click on `Proceed` and `Extend Object` to finish the UNIX extension for your account.

If you want to use Hardware Token for this account, don't forget to change OTP method and Token Type to LDAPOTP/TOKEN else you will have an error message like `Account Require Missing Data` when you will try to log in.

4. SELinux Configuration (Client Machine)

If you encountered some problems caused by SELinux so, then it's recommended to set SELinux to permissive mode.

> **Note NSCD**
>
> You have to restart nscd service if you disable SELinux configuration after having configured SpanKey Client. SELinux policies are loaded until restart each service where SELinux is configured.

For RedHat/CentOS 6:

```
bash-4.1# vi /etc/selinux/config
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

For RedHat/CentOS 7:

```
bash-4.1# vi /etc/sysconfig/selinux
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

For Debian:

By default, SELinux is not installed and configured on Debian distributions. Look the following link to have more information about SELinux on Debian.

5. SpanKey Client Setup (Client Machine)

When the spankey_client package is installed, a configuration script is available to configure it. To execute this script, you just have to run `/opt/spankey/bin/setup` in a shell prompt. The configuration of spankey client starting…

```
root@ubuntu18client:/home/ubuntu18-client# /opt/spankey/bin/setup
Enter one of your running WebADM server IP or hostname: 192.168.3.131
Detected hostname is 'ubuntu18client'. Would you like to use it as client id (y/n)? [N]
Do you want to enable SpanKey Client for OpenSSH server (y/n)? [N]
Do you want to enable SpanKey Client NSS plugin (y/n)? [Y]
Do you want to register SpanKey Client logrotate script (y/n)? [Y]
Do you want SpanKey Client to be automatically started at boot (y/n)? [Y]

Primary OpenOTP service URL is: 'https://192.168.3.131:8443/spankey/'
Secondary OpenOTP service URL is: 'NONE'.
Use 'ubuntu18client' as client id: No
Enable SpanKey Client for OpenSSH server: No
Enable SpanKey Client NSS plugin: Yes
Register SpanKey Client logrotate script: Yes
SpanKey Client must be automatically started at boot: Yes

Do you confirm (y/n)?: y

Applying SpanKey Client setting from default configuration files... Ok
Retrieving WebADM CA certificate from host '192.168.3.131'... Ok
The setup needs now to request a signed 'SpanKey' client certificate.
This request should show up as pending in your WebADM interface and an administrator
must accept it.
Waiting for approbation... Ok
```

At this step, you have to log in on the WebADM Admin GUI to approve the SSL Certificate Request in pending…

```
Updating file '/etc/nsswitch.conf'... Ok
Updating file '/etc/pam.d/common-account'...  Ok
Registering SpanKey Client service... Ok
Adding logrotate script... Ok

SpanKey Client has successfully been setup.

IMPORTANT: Do not forget to perform the following action before you exit this session:
 - Start SpanKey (/opt/spankey/bin/spankey start)
 - Restart 'nscd'
```

That's it for SpanKey client, we just use it for the NSS part so configuration is easy.

⚠ Note Debian 6

For Debian 6, you have to configure your WebADM/SpanKey Server(s) without SSL because it's not supporting by the old version of Debian. To do it, you can edit /etc/spankey/spankey.conf file. To work with SSL, you must download the source file of pam_openotp and compile it directly on the client machine.

Note: In the above example, we selected « No » to enable SpanKey for OpenSSH server because in our case we'll use SpanKey only for the NSS part. SpanKey for OpenSSH is a separate product, used in a normal way as an SSH Key Management Service requiring an enterprise license (beyond 5 managed servers). To find more information on SpanKey, please visit RCDevs | SpanKey website.

6. PAM OpenOTP Setup (Client Machine)

The configuration of the OpenOTP client is very easy. You just have to run the following command in a shell:

```
root@ubuntu18client:/home/ubuntu18-client# /usr/bin/openotp_setup
This is the configuration tool for RCDevs PAM module.
It will configure WebADM Server URL(s), SSH helper and NSS.

Enter WebADM server IP or hostname [localhost]: 192.168.3.131
Found one server URL: https://192.168.3.131:8443/openotp/
Retrieving WebADM CA certificate... Ok
Do you want PAM module to auto-create home directories ([y]/n)?:
y
Do you want to keep local password authentication as a fallback to OpenOTP? ([y]/n)?:
y
Do you want to activate PAM OpenOTP for ssh ([y]/n)?:
y
Do you want to activate PAM OpenOTP for graphical login with  lightdm ([y]/n)?:
y

Auto-create home directories: Yes
Keep local password authentication as a fallback: Yes
Activate PAM OpenOTP for ssh: Yes

Do you confirm ([y]/n)?:
y

Updating /etc/openotp/openotp.conf... Ok
Updating /etc/ssh/sshd_config... Ok
Updating /etc/pam.d/sshd... OK
Synchronizing state of ssh.service with SysV service script with /lib/systemd/systemd-
sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable ssh
Synchronizing state of nscd.service with SysV service script with /lib/systemd/systemd-
sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable nscd

PAM OpenOTP has been succesfully configured.
```

Setup for PAM OpenOTP is now finished. During the setup, we can automatically configure PAM OpenOTP for OpenSSH but we will show in the next section, the required configuration for OpenSSH.

⚠ Note Debian 6

See Note in Chapter 4. SpanKey Client.

7. OpenSSH Server Configuration (Client Machine)

For SSHd, you can keep the default configuration on each UNIX distribution. You just have to edit this file `/etc/ssh/sshd_config` and adjust the following settings:

```
ChallengeResponseAuthentication yes
UsePAM yes
```

> ⚠ Note
>
> These settings are already configured with the PAM_OpenOTP setup.

Restart OpenSSH server to apply the new configuration.

```
root@ubuntu18client:/home/ubuntu18-client# systemctl restart sshd
```

8. PAM Configuration for OpenOTP (Client Machine)

8.1 RedHat & CentOS 6/7 Distributions

All of these scenarios used LDAP accounts. The option client_id in these different configuration files are used to point to a client policy available in WebADM. If you need more information about client policy, please refer to Client Policy Guide. This setting is not mandatory, so you can remove it if you don't use it.

Scenario 1: PAM OpenOTP for Services (SSH, FTP …)

To configure UNIX services with OpenOTP authentication, you have to edit different files available in `/etc/pam.d/<service>`.

The following example working for SSH:

```
bash-4.1# vi /etc/pam.d/sshd
```

> ⚠ Note
>
> These files should be already configured with the PAM_OpenOTP setup because we answered yes to configure OpenSSH server during PAM_OpenOPT setup.

```
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so  client_id="SSH"
auth       required     pam_deny.so
account    required     pam_permit.so
session    required     pam_permit.so

account    required     pam_nologin.so
password   include      password-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
# pam_selinux.so open should only be followed by sessions to be executed in the user
context
session    required     pam_selinux.so open env_params
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      password-auth
```

Configuration is done for OpenSSH. You are now able to log in through SSH tunnel with your LDAP credential and OTP password.

Test:

```
[yoann@iMac ~]$ ssh Administrateur@192.168.3.69
Password: xxxxxxx
Enter your TOKEN password: 043792
-bash-4.1$ whoami
Administrateur
-bash-4.1$
```

Scenario 2: PAM OpenOTP for the Local Login (Console Login or Through VMWare Interface)

To configure the local login with OpenOTP (through VMWare interface for instance) you have to configure the file `/etc/pam.d/login`.

```
bash-4.1# vi /etc/pam.d/login
```

```
#%PAM-1.0
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so client_id="CONSOLE"
auth       required     pam_deny.so

account    required     pam_permit.so
session    required     pam_permit.so
account    required     pam_nologin.so
password   include      system-auth
# pam_selinux.so close should be the first session rule
session    required     pam_selinux.so close
session    required     pam_loginuid.so
session    optional     pam_console.so
# pam_selinux.so open should only be followed by sessions to be executed in the user
context
session    required     pam_selinux.so open
session    required     pam_namespace.so
session    optional     pam_keyinit.so force revoke
session    include      system-auth
-session   optional     pam_ck_connector.so
```

Test:

Scenario 3: PAM OpenOTP for SUDO

In this part, we will configure sudo to use OpenOTP. Switching user using sudo requires the necessary authorizations. These authorizations can be set by the root user and edited in `/etc/sudoers`. See UNIX documentation to edit it.

Here, we will edit `/etc/pam.d/sudo` to have a One-Time Password when users execute a sudo command.

```
bash-4.1# vi /etc/pam.d/sudo
```

```
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so client_id="sudo"
auth       required     pam_deny.so
account    required     pam_permit.so
session    required     pam_permit.so
password   include      system-auth
session    optional     pam_keyinit.so revoke
session    required     pam_limits.so
```

So, as said before, user `Administrateur` must have permissions to execute sudo command.

Test:

8.2 Debian 6/7 Distributions

Scenario 1 PAM OpenOTP for Services (SSH, FTP …)

To configure UNIX services with OpenOTP authentication, you have to edit the different file available in `/etc/pam.d/<service>`.

The following example works for SSH:

```
bash-4.1# vi /etc/pam.d/sshd
```

```
# PAM configuration for the Secure Shell service

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
auth       required     pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
auth       required     pam_env.so envfile=/etc/default/locale

# Standard Un*x authentication.
#@include common-auth
auth    sufficient   pam_unix.so
auth     sufficient   pam_openotp.so client_id="SSH"
auth     required     pam_deny.so


# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
# account  required     pam_access.so

# Standard Un*x authorization.
@include common-account

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
session    optional     pam_motd.so # [1]

# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Set up SELinux capabilities (need modified pam)
# session  required     pam_selinux.so multiple

# Standard Un*x password updating.
@include common-password
```

Configuration is done for OpenSSH. You are now able to log in through SSH tunnel with your LDAP credential and OTP password.

Scenario 2: PAM OpenOTP for the Local Login (Console or Through VMWare Interface)

```
bash-4.1# vi /etc/pam.d/login
```

```
#
# The PAM configuration file for the Shadow `login' service
#

# Enforce a minimal delay in case of failure (in microseconds).
# (Replaces the `FAIL_DELAY' setting from login.defs)
# Note that other modules may require another minimal delay. (for example,
# to disable any delay, you should add the nodelay option to pam_unix)
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so client_id="CONSOLE"
auth       required     pam_deny.so
auth       optional     pam_faildelay.so  delay=3000000

# Outputs an issue file prior to each login prompt (Replaces the
# ISSUE_FILE option from login.defs). Uncomment for use
# auth       required   pam_issue.so issue=/etc/issue

# Disallows root logins except on tty's listed in /etc/securetty
# (Replaces the `CONSOLE' setting from login.defs)
#
# With the default control of this module:
#   [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
# root will not be prompted for a password on insecure lines.
# if an invalid username is entered, a password is prompted (but login
# will eventually be rejected)
#
# You can change it to a "requisite" module if you think root may mis-type
# her login and should not be prompted for a password in that case. But
# this will leave the system as vulnerable to user enumeration attacks.
#
# You can change it to a "required" module if you think it permits to
# guess valid user names of your system (invalid user names are considered
# as possibly being root on insecure lines), but root passwords may be
# communicated over insecure lines.
auth [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
pam_securetty.so

# Disallows other than root logins when /etc/nologin exists
# (Replaces the `NOLOGINS_FILE' option from login.defs)
auth       requisite  pam_nologin.so

# SELinux needs to be the first session rule. This ensures that any
# lingering context has been cleared. Without out this it is possible
# that a module could execute code in the wrong domain.
# When the module is present, "required" would be sufficient (When SELinux
```

```
# is disabled, this returns success.)
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
close

# This module parses environment configuration file(s)
# and also allows you to use an extended config
# file /etc/security/pam_env.conf.
#
# parsing /etc/environment needs "readenv=1"
session       required   pam_env.so readenv=1
# locale variables are also kept into /etc/default/locale in etch
# reading this file *in addition to /etc/environment* does not hurt
session       required   pam_env.so readenv=1 envfile=/etc/default/locale

# Standard Un*x authentication.
@include common-auth

# This allows certain extra groups to be granted to a user
# based on things like time of day, tty, service, and user.
# Please edit /etc/security/group.conf to fit your needs
# (Replaces the `CONSOLE_GROUPS' option in login.defs)
auth       optional   pam_group.so

# Uncomment and edit /etc/security/time.conf if you need to set
# time restrainst on logins.
# (Replaces the `PORTTIME_CHECKS_ENAB' option from login.defs
# as well as /etc/porttime)
# account    requisite  pam_time.so

# Uncomment and edit /etc/security/access.conf if you need to
# set access limits.
# (Replaces /etc/login.access file)
# account  required       pam_access.so

# Sets up user limits according to /etc/security/limits.conf
# (Replaces the use of /etc/limits in old login)
session    required   pam_limits.so

# Prints the last login info upon succesful login
# (Replaces the `LASTLOG_ENAB' option from login.defs)
session    optional   pam_lastlog.so

# Prints the message of the day upon succesful login.
# (Replaces the `MOTD_FILE' option in login.defs)
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional   pam_motd.so  motd=/run/motd.dynamic
session    optional   pam_motd.so

# Prints the status of the user's mailbox upon succesful login
# (Replaces the `MAIL_CHECK_ENAB' option from login.defs).
#
```

```
# This also defines the MAIL environment variable
# However, userdel also needs MAIL_DIR and MAIL_FILE variables
# in /etc/login.defs to make sure that removing a user
# also removes the user's mail spool file.
# See comments in /etc/login.defs
session    optional   pam_mail.so standard

# Standard Un*x account and session
@include common-account
@include common-session
@include common-password

# SELinux needs to intervene at login time to ensure that the process
# starts in the proper default security context. Only sessions which are
# intended to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
open
# When the module is present, "required" would be sufficient (When SELinux
# is disabled, this returns success.)
```

Scenario 3: PAM OpenOTP for SUDO

In this part, we will configure sudo to use OpenOTP. Switching user using sudo requires the necessary authorizations. These authorizations can be set by the root user and edited in `/etc/sudoers`. See UNIX documentation to edit it.

Here, we will edit `/etc/pam.d/sudo` to prompt a One-Time Password when users execute a sudo command.

```
bash-4.1# vi /etc/pam.d/sudo
```

```
#%PAM-1.0
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so client_id="sudo"
auth       required     pam_deny.so


#@include common-auth
@include common-account
@include common-session-noninteractive

#session required pam_permit.so
#session required pam_limits.so
```

So, as said before, user `Administrateur` must have the permissions to execute sudo command.

Test:

8.3 Debian 8 and Later Distributions

On Debian 8, some configuration files are different from the previous version of Debian. See below, the configuration files for the different scenario on Debian 8.

Scenario 1: PAM OpenOTP for Services (SSH, FTP…)

Here we'll show how to configure the UNIX services with OpenOTP authentication.

```
admin:~ lo$ ssh user_spankey@192.168.3.130
Password:
Enter your TOKEN password: 282027
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-43-generic x86_64)
...
Last login: Fri Jan  4 14:44:06 2019 from 192.168.3.233
user_spankey@ubuntu18client:~$
```

To configure UNIX services with OpenOTP authentication, you have to edit the different files available in `/etc/pam.d/<service>`.

The following example works for SSH:

```
root@ubuntu18client:/home/ubuntu18-client# vi /etc/pam.d/sshd
```

```
# PAM configuration for the Secure Shell service

# Standard Un*x authentication.
@include openotp-auth

# Disallow non-root logins when /etc/nologin exists.
account    required     pam_nologin.so

# Uncomment and edit /etc/security/access.conf if you need to set complex
# access limits that are hard to express in sshd_config.
# account  required     pam_access.so

# Standard Un*x authorization.
@include common-account

# SELinux needs to be the first session rule.  This ensures that any
# lingering context has been cleared.  Without this it is possible that a
# module could execute code in the wrong domain.
session [success=ok ignore=ignore module_unknown=ignore default=bad]
```

```
pam_selinux.so close

# Set the loginuid process attribute.
session    required     pam_loginuid.so

# Create a new session keyring.
session    optional     pam_keyinit.so force revoke

# Standard Un*x session setup and teardown.
@include common-session

# Print the message of the day upon successful login.
# This includes a dynamically generated part from /run/motd.dynamic
# and a static (admin-editable) part from /etc/motd.
session    optional     pam_motd.so  motd=/run/motd.dynamic
session    optional     pam_motd.so noupdate

# Print the status of the user's mailbox upon successful login.
session    optional     pam_mail.so standard noenv # [1]

# Set up user limits from /etc/security/limits.conf.
session    required     pam_limits.so

# Read environment variables from /etc/environment and
# /etc/security/pam_env.conf.
session    required     pam_env.so # [1]
# In Debian 4.0 (etch), locale-related environment variables were moved to
# /etc/default/locale, so read that as well.
session    required     pam_env.so user_readenv=1 envfile=/etc/default/locale

# SELinux needs to intervene at login time to ensure that the process starts
# in the proper default security context.  Only sessions which are intended
# to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad]
pam_selinux.so open

# Standard Un*x password updating.
@include common-password
```

```
root@ubuntu18client:/home/ubuntu18-client# vi /etc/pam.d/openotp-auth
```

```
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so
auth       required     pam_deny.so
```

Scenario 2: PAM OpenOTP for the Local Login (Console or Through VMWare Interface)

Here we'll show how to configure the local terminal login for example through the VMWare Interface with OpenOTP. You have to configure the following file `/etc/pam.d/login`.

```
root@ubuntu18client:/home/ubuntu18-client# vi /etc/pam.d/login
```

```
#
# The PAM configuration file for the Shadow `login' service
#
auth       sufficient   pam_openotp.so client_id="CONSOLE"

# Enforce a minimal delay in case of failure (in microseconds).
# (Replaces the `FAIL_DELAY' setting from login.defs)
# Note that other modules may require another minimal delay. (for example,
# to disable any delay, you should add the nodelay option to pam_unix)
auth       optional   pam_faildelay.so  delay=3000000

# Outputs an issue file prior to each login prompt (Replaces the
# ISSUE_FILE option from login.defs). Uncomment for use
# auth       required   pam_issue.so issue=/etc/issue

# Disallows root logins except on tty's listed in /etc/securetty
# (Replaces the `CONSOLE' setting from login.defs)
#
# With the default control of this module:
#   [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
# root will not be prompted for a password on insecure lines.
# if an invalid username is entered, a password is prompted (but login
# will eventually be rejected)
#
# You can change it to a "requisite" module if you think root may mis-type
# her login and should not be prompted for a password in that case. But
# this will leave the system as vulnerable to user enumeration attacks.
#
# You can change it to a "required" module if you think it permits to
# guess valid user names of your system (invalid user names are considered
# as possibly being root on insecure lines), but root passwords may be
# communicated over insecure lines.
auth [success=ok new_authtok_reqd=ok ignore=ignore user_unknown=bad default=die]
pam_securetty.so

# Disallows other than root logins when /etc/nologin exists
# (Replaces the `NOLOGINS_FILE' option from login.defs)
auth       requisite  pam_nologin.so

# SELinux needs to be the first session rule. This ensures that any
# lingering context has been cleared. Without this it is possible
# that a module could execute code in the wrong domain.
# When the module is present, "required" would be sufficient (When SELinux
# is disabled, this returns success.)
```

```
# is disabled, this returns success.)
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
close

# Sets the loginuid process attribute
session    required     pam_loginuid.so

# SELinux needs to intervene at login time to ensure that the process
# starts in the proper default security context. Only sessions which are
# intended to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
open
# When the module is present, "required" would be sufficient (When SELinux
# is disabled, this returns success.)

# This module parses environment configuration file(s)
# and also allows you to use an extended config
# file /etc/security/pam_env.conf.
#
# parsing /etc/environment needs "readenv=1"
session       required   pam_env.so readenv=1

# Standard Un*x authentication.
@include common-auth

# This allows certain extra groups to be granted to a user
# based on things like time of day, tty, service, and user.
# Please edit /etc/security/group.conf to fit your needs
# (Replaces the `CONSOLE_GROUPS' option in login.defs)
auth       optional   pam_group.so

# Uncomment and edit /etc/security/time.conf if you need to set
# time restraint on logins.
# (Replaces the `PORTTIME_CHECKS_ENAB' option from login.defs
# as well as /etc/porttime)
# account    requisite  pam_time.so

# Uncomment and edit /etc/security/access.conf if you need to
# set access limits.
# (Replaces /etc/login.access file)
# account  required       pam_access.so

# Sets up user limits according to /etc/security/limits.conf
# (Replaces the use of /etc/limits in old login)
session    required   pam_limits.so

# Prints the last login info upon successful login
# (Replaces the `LASTLOG_ENAB' option from login.defs)
session    optional   pam_lastlog.so

# Prints the message of the day upon successful login.
# (Replaces the `MOTD_FILE' option in login.defs)
# This includes a dynamically generated part from /run/motd.dynamic
```

```
# and a static (admin-editable) part from /etc/motd.
session    optional   pam_motd.so motd=/run/motd.dynamic
session    optional   pam_motd.so noupdate

# Prints the status of the user's mailbox upon successful login
# (Replaces the `MAIL_CHECK_ENAB' option from login.defs).
#
# This also defines the MAIL environment variable
# However, userdel also needs MAIL_DIR and MAIL_FILE variables
# in /etc/login.defs to make sure that removing a user
# also removes the user's mail spool file.
# See comments in /etc/login.defs
session    optional   pam_mail.so standard

# Create a new session keyring.
session    optional   pam_keyinit.so force revoke

# Standard Un*x account and session
@include common-account
@include common-session
@include common-password
```

Scenario 3: PAM OpenOTP for SUDO

In this part, we will configure sudo to use OpenOTP.

```
user_spankey@ubuntu18client:~$ whoami
user_spankey
user_spankey@ubuntu18client:~$ sudo su
[sudo] password for user_spankey:
Enter your TOKEN password: 745487
root@ubuntu18client:/home/user_spankey# whoami
root
```

Switching the user to use sudo requires the necessary authorizations. These authorizations can be set by the root user by editing the `/etc/sudoers` file. See UNIX documentation to edit it.

First, we'll add the user (user_spankey) to `/etc/sudoers` with the following command:

```
root@ubuntu18client:/home/ubuntu18-client# addgroup user_spankey sudo
Adding user `user_spankey' to group `sudo' ...
Adding user user_spankey to group sudo
Done.
```

Here, we will edit `/etc/pam.d/sudo` to prompt a One-Time Password when users execute a sudo command.

```
root@ubuntu18client:/home/ubuntu18-client# vi /etc/pam.d/sudo
```

```
#%PAM-1.0

auth sufficient pam_openotp.so client_id="sudo"
session    required   pam_env.so readenv=1 user_readenv=0
session    required   pam_env.so readenv=1 envfile=/etc/default/locale user_readenv=0
@include common-auth
@include common-account
@include common-session-noninteractive
```

8.4 PAM OpenOTP for Login to the Desktop Environment

First, you have to determine which desktop environment you are running lightdm, Gnome desktop… In this documentation, we will show you how to configure PAM OpenOTP login for these last 3 interfaces.

8.4.1 Ubuntu 16.04 LTS - lightdm

For Ubuntu, the default graphical interface is lightdm. To authorize the user to enter his own username, you have to edit the following file:

```
vi /usr/share/lightdm/lightdm.conf.d/50-ubuntu.conf
```

And add the following line:

```
greeter-show-manual-login=true
```

You can now reboot your machine and you will be able on the next login to enter your username manually.

Now, go in `/etc/pam.d/` folder and look if the openotp-auth file is present. Normally it should be here after the openotp_setup script.

After that, you can edit the file `/etc/pam.d/lightdm` and you should have something like below:

```
#%PAM-1.0
auth    requisite       pam_nologin.so
auth    sufficient      pam_succeed_if.so user ingroup nopasswdlogin

@include openotp-auth
auth    optional        pam_gnome_keyring.so
auth    optional        pam_kwallet.so
auth    optional        pam_kwallet5.so
@include common-account
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
close
session required        pam_loginuid.so
session required        pam_limits.so
@include common-session
session [success=ok ignore=ignore module_unknown=ignore default=bad] pam_selinux.so
open
session optional        pam_gnome_keyring.so auto_start
session optional        pam_kwallet.so auto_start
session optional        pam_kwallet5.so auto_start
session required        pam_env.so readenv=1
session required        pam_env.so readenv=1 user_readenv=1 envfile=/etc/default/locale
@include common-password
```

This is the default file, we only change `@include common-auth` by `@include openotp-auth` on line 5.

Configuration is done, you are now able to log in to your desktop with an OTP.

8.4.2 Debian 9 - Gnome (GDM)

For GDM, the only file that you have to edit is: `/etc/pam.d/gdm-password`. This file should be like below:

```
#%PAM-1.0
auth    requisite       pam_nologin.so
auth    required        pam_succeed_if.so user != root quiet_success
@include openotp-auth
auth    optional        pam_gnome_keyring.so
@include common-account
# SELinux needs to be the first session rule. This ensures that any
# lingering context has been cleared. Without this it is possible
# that a module could execute code in the wrong domain.
session [success=ok ignore=ignore module_unknown=ignore default=bad]
pam_selinux.so close
session required        pam_loginuid.so
# SELinux needs to intervene at login time to ensure that the process
# starts in the proper default security context. Only sessions which are
# intended to run in the user's context should be run after this.
session [success=ok ignore=ignore module_unknown=ignore default=bad]
pam_selinux.so open
session optional        pam_keyinit.so force revoke
session required        pam_limits.so
session required        pam_env.so readenv=1
session required        pam_env.so readenv=1 envfile=/etc/default/locale
@include common-session
session optional        pam_gnome_keyring.so auto_start
@include common-password
```

This is the default file, we only change `@include common-auth` by `@include openotp-auth` on line 4.

Configuration is done, you are now able to login on the Gnome desktop with an OTP:

8.4.3 CentOS 7 - Gnome (GDM)

For GDM, the only file that you have to edit is: `/etc/pam.d/gdm-password`. This file should be like below:

```
auth     [success=done ignore=ignore default=bad] pam_selinux_permit.so
auth       substack     openotp-auth
@include openotp-auth
auth        optional      pam_gnome_keyring.so
auth        include       postlogin

account     required      pam_nologin.so
account     include       password-auth

password    substack       password-auth
-password   optional       pam_gnome_keyring.so use_authtok

session     required      pam_selinux.so close
session     required      pam_loginuid.so
session     optional      pam_console.so
session     required      pam_selinux.so open
session     optional      pam_keyinit.so force revoke
session     required      pam_namespace.so
session     include       password-auth
session     optional      pam_gnome_keyring.so auto_start
session     include       postlogin
```

This is the default file, we change it by disabling `#auth substack openotp-auth` and `#@include openotp-auth`. Finally, adding the following:

```
cat /etc/pam.d/openotp-auth
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so
auth       required     pam_deny.so
```

```
auth     [success=done ignore=ignore default=bad] pam_selinux_permit.so
#auth       substack     openotp-auth
#@include openotp-auth
auth       required     pam_env.so
auth       sufficient   pam_unix.so
auth       sufficient   pam_openotp.so
auth       required     pam_deny.so

auth        optional      pam_gnome_keyring.so
auth        include       postlogin

account     required      pam_nologin.so
account     include       password-auth

password    substack       password-auth
-password   optional       pam_gnome_keyring.so use_authtok

session     required      pam_selinux.so close
session     required      pam_loginuid.so
session     optional      pam_console.so
session     required      pam_selinux.so open
session     optional      pam_keyinit.so force revoke
session     required      pam_namespace.so
session     include       password-auth
session     optional      pam_gnome_keyring.so auto_start
session     include       postlogin
```

Configuration is done, you are now able to login on the Gnome desktop with an OTP:

9. Troubleshooting

There are many files that you can check to troubleshoot the Linux client and WebADM/OpenOTP/SpanKey servers.

9.1 WebADM/OpenOTP/SpanKey Servers

WebADM/OpenOTP has a transaction log that records all requests/responses in the following file:

```
bash-4.1# cat /opt/webadm/logs/webadm.log
```

Typical logs of an authentication success using SSH and PAM_OpenOTP:

```
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  New openotpSimpleLogin SOAP
request
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Username: Administrateur
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Password: xxxxxxxx
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Client ID: SSH
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Source IP: 10.0.3.22
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Options: -U2F
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Enforcing client policy: SSH
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Registered openotpSimpleLogin
request
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Resolved LDAP user:
CN=Administrateur,CN=Users,DC=yorcdevs,DC=com
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Resolved LDAP groups:
propri\xc3\xa9taires cr\xc3\xa9ateurs de la strat\xc3\xa9gie de groupe,admins du
domaine,administrateurs de l\xe2\x80\x99entreprise,administrateurs du
sch\xc3\xa9ma,administrateurs,utilisateurs du bureau \xc3\xa0 distance,groupe de
r\xc3\xa9plication dont le mot de passe rodc est refus\xc3\xa9
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Started transaction lock for
user
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found user language: EN
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found 3 user certificate
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found 37 user settings:
LoginMode=LDAPMFA,OTPType=TOKEN,OTPLength=6,ChallengeMode=Yes,ChallengeTimeout=90,Challer
1:HOTP-SHA1-6:QN06-
T1M,SMSType=Normal,SMSMode=Ondemand,MailMode=Ondemand,LastOTPTime=300,ListChallengeMode=S

[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found 12 user data:
LoginCount,RejectCount,LastOTP,ListInit,ListState,TokenType,TokenKey,TokenState,TokenID,D

[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found 1 registered OTP token
(TOTP)
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Requested login factors: LDAP
& OTP
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  LDAP password Ok
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Challenge required
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Started OTP challenge session
of ID PaS3WXe2HDJFz0st valid for 90 seconds
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Sent challenge response
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  New openotpChallenge SOAP
request
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Username: Administrateur
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > Session: PaS3WXe2HDJFz0st
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  > OTP Password: xxxxxx
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Enforcing client policy: SSH
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Registered openotpChallenge
request
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Found challenge session
started 2017-02-03 15:54:30
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Started transaction lock for
```

```
user
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  TOTP password Ok (token #1)
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Updated user data
[2017-02-03 15:54:30] [192.168.3.134] [OpenOTP:3MJAB3KR]  Sent success response
```

Typical logs of an authentication failure caused by WebADM configuration. `Challenge Mode Supported` should be configured to `Yes` either in OpenOTP Application settings or in the sudo Client Policy settings.

```
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  New openotpSimpleLogin SOAP
request
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  > Username: Administrateur
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  > Password: xxxxxxxx
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  > Client ID: sudo
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  > Source IP: 10.0.3.21
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Options: -U2F
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Enforcing client policy: sudo
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Registered openotpSimpleLogin
request
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Resolved LDAP user:
CN=Administrateur,CN=Users,DC=yorcdevs,DC=com
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Resolved LDAP groups:
propri\xc3\xa9taires cr\xc3\xa9ateurs de la strat\xc3\xa9gie de groupe,admins du
domaine,administrateurs de l\xe2\x80\x99entreprise,administrateurs du
sch\xc3\xa9ma,administrateurs,utilisateurs du bureau \xc3\xa0 distance,groupe de
r\xc3\xa9plication dont le mot de passe rodc est refus\xc3\xa9
[2017-02-03 13:26:41] [192.168.3.60] [OpenOTP:7UERIOQE]  Started transaction lock for
user
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Found user language: EN
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Found 3 user certificate
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Found 37 user settings:
LoginMode=LDAPMFA,OTPType=TOKEN,OTPLength=6,ChallengeMode=Yes,ChallengeTimeout=90,Challer
1:HOTP-SHA1-6:QN06-
T1M,SMSType=Normal,SMSMode=Ondemand,MailMode=Ondemand,LastOTPTime=300,ListChallengeMode=S

[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Found 12 user data:
LoginCount,RejectCount,LastOTP,ListInit,ListState,TokenType,TokenKey,TokenState,TokenID,D

[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Challenge mode disabled
(assuming concatened passwords)
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Found 1 registered OTP token
(TOTP)
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Requested login factors: LDAP
& OTP
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  LDAP password Ok
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Updated user data
[2017-02-03 13:26:44] [192.168.3.60] [OpenOTP:7UERIOQE]  Sent failure response
```

9.2 SpanKey Client

To know if SpanKey client works properly, you can run the following command on your client:

```
bash-4.1# getent passwd
```

This command must return Local and LDAP account (Extended to UNIX in WebADM).

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
Debian-exim:x:104:109::/var/spool/exim4:/bin/false
messagebus:x:105:110::/var/run/dbus:/bin/false
statd:x:106:65534::/var/lib/nfs:/bin/false
sshd:x:107:65534::/var/run/sshd:/usr/sbin/nologin
test:x:1000:1000::/home/test:/bin/bash
Administrateur:x:1100:100::/home/administrateur:/bin/bash
yo:x:1101:100::/home/yo:/bin/sh
```

You should see a UNIX extended LDAP account in the result of the `getent passwd` command:

```
test:x:1000:1000:test::/home/test:/bin/bash
Administrateur:x:1100:100::/home/administrateur:/bin/bash
yo:x:1101:100::/home/yo:/bin/sh
```

If this command doesn't return your LDAP Accounts, please check the firewall configuration and SpanKey's configuration URLs in

/etc/spankey/spankey.conf. You can also try to restart the nscd service and check the SELinux configuration.

On Unix client, you can see the logs of the getent command in `/var/log/messages`:

```
Feb  3 15:33:40 debian8 spankey[2043]: RCDevs SpanKey NSS Plugin version 1.0.2-3 loaded
```

If this log doesn't appear when you call the getent command, SpanKey is not installed correctly. Try to reinstall it.

In WebADM logs, you can see the logs for the getent command too, getent call SpanKey module:

```
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] New spankeyNSSList SOAP
request
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] > Database: user
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] > Client ID: SSH
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] Enforcing client policy: SSH
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] Registered spankeyNSSList
request
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] Found 2 posix users
[2017-02-03 16:48:15] [192.168.3.134] [SpanKey:VOK85UQY] Sent success response
```

9.3 CentOS & Debian

Typical logs of an authentication success:

For CentOS:

```
bash-4.1# cat /var/log/secure
```

```
Feb  3 16:24:30 centos7 openotp[2132]: PAM Module for OpenOTP version 1.0.12 starting
Feb  3 16:24:30 centos7 openotp[2132]: Server URLs: https://192.168.3.55:8443/openotp/
Feb  3 16:24:30 centos7 openotp[2132]: Server Policy: Ordered
Feb  3 16:24:30 centos7 openotp[2132]: Domain name: [None]
Feb  3 16:24:30 centos7 openotp[2132]: Client id: SSH
Feb  3 16:24:30 centos7 openotp[2132]: Challenge suffix: :
Feb  3 16:24:30 centos7 openotp[2132]: User settings: [None]
Feb  3 16:24:30 centos7 openotp[2132]: Cert file: [None]
Feb  3 16:24:30 centos7 openotp[2132]: Cert password: [None]
Feb  3 16:24:30 centos7 openotp[2132]: CA file: [None]
Feb  3 16:24:30 centos7 openotp[2132]: SOAP timeout: [Default]
Feb  3 16:24:30 centos7 openotp[2132]: Create homedirs: No
Feb  3 16:24:30 centos7 openotp[2132]: Password mode: [Default]
Feb  3 16:24:30 centos7 openotp[2132]: Password separator: [None]
Feb  3 16:24:30 centos7 openotp[2132]: OTP length: [Default]
Feb  3 16:24:30 centos7 openotp[2132]: Got user name Administrateur
Feb  3 16:24:30 centos7 openotp[2132]: Got host name 10.0.3.28
Feb  3 16:24:30 centos7 openotp[2132]: Got anyPassword ******** for user Administrateur
Feb  3 16:24:30 centos7 openotp[2132]: Sending OpenOTP SimpleLogin request for user
Administrateur
Feb  3 16:24:31 centos7 openotp[2132]: Authentication challenge for user Administrateur
Feb  3 16:24:53 centos7 openotp[2132]: Got OTP password ****** for user Administrateur
Feb  3 16:24:53 centos7 openotp[2132]: Sending OpenOTP Challenge request for user
Administrateur
Feb  3 16:24:56 centos7 openotp[2132]: Authentication succeeded for user Administrateur
```

For Debian:

```
bash-4.1# cat /var/log/auth.log
```

```
Feb  3 15:54:30 debian8 openotp[2048]: PAM Module for OpenOTP version 1.0.12 starting
Feb  3 15:54:30 debian8 openotp[2048]: Server URLs: https://192.168.3.55:8443/openotp/
Feb  3 15:54:30 debian8 openotp[2048]: Server Policy: Ordered
Feb  3 15:54:30 debian8 openotp[2048]: Domain name: [None]
Feb  3 15:54:30 debian8 openotp[2048]: Client id: SSH
Feb  3 15:54:30 debian8 openotp[2048]: Challenge suffix: :
Feb  3 15:54:30 debian8 openotp[2048]: User settings: [None]
Feb  3 15:54:30 debian8 openotp[2048]: Cert file: [None]
Feb  3 15:54:30 debian8 openotp[2048]: Cert password: [None]
Feb  3 15:54:30 debian8 openotp[2048]: CA file: [None]
Feb  3 15:54:30 debian8 openotp[2048]: SOAP timeout: [Default]
Feb  3 15:54:30 debian8 openotp[2048]: Create homedirs: No
Feb  3 15:54:30 debian8 openotp[2048]: Password mode: [Default]
Feb  3 15:54:30 debian8 openotp[2048]: Password separator: [None]
Feb  3 15:54:30 debian8 openotp[2048]: OTP length: [Default]
Feb  3 15:54:30 debian8 openotp[2048]: Got user name Administrateur
Feb  3 15:54:30 debian8 openotp[2048]: Got host name 10.0.3.22
Feb  3 15:54:30 debian8 openotp[2048]: Got anyPassword ******** for user Administrateur
Feb  3 15:54:30 debian8 openotp[2048]: Sending OpenOTP SimpleLogin request for user
Administrateur
Feb  3 15:54:31 debian8 openotp[2048]: Authentication challenge for user Administrateur
Feb  3 15:54:53 debian8 openotp[2048]: Got OTP password ****** for user Administrateur
Feb  3 15:54:53 debian8 openotp[2048]: Sending OpenOTP Challenge request for user
Administrateur
Feb  3 15:54:56 debian8 openotp[2048]: Authentication succeeded for user Administrateur
```

9.4 Name Service Cache Daemon (NSCD)

In Linux, user and group information is often cached by NSCD (Name Service Cache Daemon), this can result in failed PAM-OpenOTP login right after the installation or after creating a new user since the user is not available in the cache yet.

To resolve this issue, you can wait for the cache to be refreshed on its own, or start and stop the nscd process and to flush the NSCD cache on your server.

The exact command and configuration depend on the Linux distribution in question. These commands are a sample for CentOS 7.

To stop and start NSCD:

```
systemctl stop nscd
systemctl start nscd
```

To clear NSCD cache files, invalidate the passwd and group cache:

```
[root@centos8-client ~]# nscd --invalidate=passwd
[root@centos8-client ~]# nscd --invalidate=group
```

10. Video Tutorial for OpenSSH

Play Video on Youtube