Opkssh - SSH via OIDC

Since we have installed Authentik, let's use it to create access groups to servers via SSH. To do this, we will use the project on the page (https://github.com/openpubkey/opkssh) [https://github.com/openpubkey/opkssh].

Step 1: Configuring provider and application

First of all, let's enter the Authentik administration interface and create an ad-hoc provider (Left menu -> Applications -> Providers): it must be of type *public* and **NOT** confidential. Let's note down the client ID:

- Provider Name: opkssh-provider
- Client type: Public
- Redirect URIs:
  - http://localhost:3000/login-callback
  - http://localhost:10001/login-callback
  - http://localhost:11110/login-callback Yes, there are 3 of them.

The last item to change is the Subject mode, which is best set to *Based on the user's UUID*. The rest can be left as is.

We need to associate an application with it (Left menu -> Applications -> Applications):

- Application name: Opkssh
- Slug: opkssh
- Provider: opkssh-provider

The rest, as is.

Step 2: Group creation (at least one)

To access servers, unless you want to grant individual name-based permissions on each machine, we can enable groups of people to log in as user "X" (**who must exist on the machines as a local user**). Are you the user "Mario Rossi"? If you belong to the "ssh-admins" group, you will be able to log into the machine as its local user "X" (X can be

any system user, even root). It goes without saying that we can create as many groups as we want for this purpose: maybe the *devops* group, the *secops* group, the *dishwasher* group, depending on how granular we want to be with our host inventory. We will only create *ssh-admins*.

That said: Left menu -> Directory -> Groups and let's create *ssh-admins*. After this, top menu, Users -> Add existing user and assign the user we want to test.

WARNING: for the record, we should associate the newly created *Opkssh* application only to those entitled to SSH connections, so it might be appropriate to create an *ssh-access* group and subgroups like *ssh-admins*, *ssh-devops*, etc., to associate them with *ssh-access*. For now, I'll pass because I'm sleepy.

Step 3: Server-side installation

Let's proceed with the server-side installation:

```
wget -qO- "https://raw.githubusercontent.com/openpubkey/opkssh/main/scripts/install-linux.sh" | sudo bash
```

Let's edit the `/etc/ssh/sshd_config` file.

Replace:

```
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
```

with:

```
AuthorizedKeysCommand /usr/local/bin/opkssh verify %u %k %t
AuthorizedKeysCommandUser opksshuser
```

and restart the server with `systemctl restart ssh`.

Let's edit the `/etc/opk/providers` file so that it only contains our identity provider:

```
https://auth.example.org/application/o/opkssh/
AZEQOFmwX6qTQOG63ekDHMceSFehjkRUdtayA3yh 24h
```

where `AZEQOFmwX6qTQOG63ekDHMceSFehjkRUdtayA3yh` is the client ID created in Step 1, during provider configuration.

Let's create the user that will be used by whoever logs in via OIDC:

```
adduser oidcuser
info: Adding user `oidcuser' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Adding new group `oidcuser' (1003) ...
info: Adding new user `oidcuser' (1003) with group `oidcuser (1003)' ...
info: Creating home directory `/home/oidcuser' ...
info: Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for oidcuser
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
info: Adding new user `oidcuser' to supplemental / extra groups `users' ...:
info: Adding user `oidcuser' to group `users' ...

# usermod -aG sudo oidcuser
```

and specify that whoever belongs to the identity provider's `ssh-admins` group can log into the machine (and will enter as the user `oidcuser`):

```
# opkssh add oidcuser oidc:groups:ssh-admins
https://auth.example.org/application/o/opkssh/
2026/03/20 09:22:23 Successfully added user with email oidc:groups:ssh-admins
with principal oidcuser to the policy file
Successfully added new policy to /etc/opk/auth_id
```

Step 4: Tlog

To keep track of what the *oidcuser* does, we will use *tlog*.

```
sudo apt update && sudo apt install tlog jq -y
```

Let's configure recording:

```
sudo mkdir -p /etc/tlog
sudo tee /etc/tlog/tlog-rec-session.conf <<EOF
{
  "shell": "/bin/bash",
  "writer": "file",
  "file": {
    "path": "/var/log/tlog-ssh.log"
  }
}
EOF
```

Let's configure the player:

```
sudo tee /etc/tlog/tlog-play.conf <<EOF
{
  "reader": "file",
  "file": {
    "path": "/var/log/tlog-ssh.log"
  }
}
EOF
```

Let's prepare the log file and permissions:

```
# Create the log file and set permissions (writable by everyone, but not deletable)
sudo touch /var/log/tlog-ssh.log
sudo chmod 622 /var/log/tlog-ssh.log

# Configure the temporary directory for locks (prevents Permission Denied error)
sudo tee /etc/tmpfiles.d/tlog.conf <<EOF
d /run/tlog 1777 root root -
EOF
sudo systemd-tmpfiles --create /etc/tmpfiles.d/tlog.conf
```

Let's enable recording for the user:

```
# Add tlog to valid shells
echo "/usr/bin/tlog-rec-session" | sudo tee -a /etc/shells

# Change the user's shell (e.g. oidcuser)
sudo chsh -s /usr/bin/tlog-rec-session oidcuser
```

For the administrator

Here is a cheatsheet for an administrator:

- See how many connections there are: `sudo cat /var/log/tlog-ssh.log | jq -r '.rec' | sort | uniq | wc -l`
- List sessions (User, ID, Date): `sudo cat /var/log/tlog-ssh.log | jq -r '[.user, .rec, (.time | strftime("%Y-%m-%d %H:%M:%S"))] | @tsv' | column -tH`
- Replay the last session: `tlog-play`
- Replay a specific session: `tlog-play -M "rec=INSERT_ID_HERE"`

Step 5: Client installation

The client is what requests the token from the IdP: once obtained, it will receive the ECDSA key which will be written under *~/.ssh/* and can be used to connect to remote servers. The installation file can be retrieved from (https://github.com/openpubkey/opkssh) [https://github.com/openpubkey/opkssh], in the releases section.

Once installed, it will be sufficient, as a user, to run:

```
opkssh login --create-config
```

This will create `~/.opk/config.yml`. Let's delete its content and replace it:

```
---
default_provider: webchooser

providers:
  - alias: Dreadful
    issuer: https://auth.example.org/application/o/opkssh/
    client_id: AZEQOFmwX6qTQOG63ekDHMceSFehjkRUdtayA3yh
    scopes: openid profile email offlince_access
    access_type: offline
```

```
    prompt: consent
    redirect_uris:
      - http://localhost:3000/login-callback
      - http://localhost:10001/login-callback
      - http://localhost:11110/login-callback
```

Once again, `AZEQOFmwX6qTQOG63ekDHMceSFehjkRUdtayA3yh` is the client ID created in Step 1, during provider configuration.

At this point, for authentication, simply execute:

```
opkssh login
```

the browser will open and ask for credentials. From this moment on, we can start connecting via:

```
ssh oidcuser@server
```

[May the big mole be with you]

[EOF]