whawty.auth store sync

As the whawty.auth store is just a simple directory you may synchronise multiple instances using rsync. One way how to do this is documented here.

Introduction

The synchronisation is based on a simple master slave system. The master host is an app running on one machine which might be configured to do local upgrades (see below). One or more slaves use a systemd.timer, rsync and ssh to synchronise the local file store with the one on the (remote) master host. This is quite straight forward and a sample setup is documented below

context upgrades

The whawty.auth app can be configured to automatically upgrade passwords when users authenticate against it. For this to work the storage backend compares the current default context-id, as set by the store configuration, with the one which was used to generate the current hash. If the context-ids differ it marks the hash as upgradeable If the authentication was successful the app now does one of the following:

- do nothing: no upgrade will be done, the files will stay untouched
- local upgrades: do an update operation on the local store using the old password.
- remote upgrades: do an update operation on a remote host using the web api of this host.

The first 2 steps are quite self-explanatory. For the third to work the slave host must be able to reach the masters web API. The uri for the update endpoint can be configured using the –do-upgrades parameter. Mind that the app has no support for https and it is strongly recommended to have a SSL-enabling reverse proxy in front of it.

The remote upgrades are opportunistic which means that a failed update call will silently be ignored. This is so that a temporary communcations problem between master and slave don't slows down local authentication requests. There is also a limit for concurrent remote upgrade calls to the master.

Setup

Master

This assumes you have whawty.auth app as user whawty-auth. The store directory is at '/var/lib/whawty/auth/store'. '/var/lib/whawty/auth' is the home of the user whawty-auth. In order to enable automatic upgrades the web api should be enabled and the app should be configured to do local upgrades. Also a SSL-enabling reverse proxy is configured to forward requests to https://whawty-auth-master.example.com/api/ to the app. Besides that ssh must be running on that host and rsync needs to be installed.

Copy the file 'auth-rsyncd.conf' to '/etv/whawty/'. Then run the following commands to create an 'authorized_keys' file for the user whawty-auth.

```
# mkdir /var/lib/whawty/auth/.ssh
# chmod 700 /var/lib/whawty/auth/.ssh
# touch /var/lib/whawty/auth/.ssh/authorized_keys
# chmod 600 /var/lib/whawty/auth/.ssh/authorized_keys
# chown -R  whawty-auth:whawty-auth /var/lib/whawty/auth/.ssh
```

The authorized keys file should contain one line of the following form for each slave:

```
command="rsync --server --config /etc/whawty/auth-rsyncd.conf --daemon .",no-X11-forwarding,
```

Slave

This assumes the slave runs as the same user and uses the same store base directory as the master. Us the following commands to create a .ssh directory:

```
# mkdir /var/lib/whawty/auth/.ssh
# chmod 700 /var/lib/whawty/auth/.ssh
# chown whawty-auth:whawty-auth /var/lib/whawty/auth/.ssh
# sudo -u whawty-auth ssh-keygen -t ed25519
```

As user 'whawty-auth' open/create the file '/var/lib/whawty/auth/.ssh/config' and add the following to it:

```
Host whawty-auth-master
  HostName <hostname or IP of master>
  IdentityFile ~/.ssh/id_ed25519
  IdentitiesOnly yes
```

Use the contents of '/var/lib/whawty/auth/.ssh/id_ed25519.pub' to add the entry to the 'authorized_keys' file of the master as documented above. You should now be able to sync password hashes from the master using the following command:

```
# sudo -u whawty-auth rsync -rlptv --delete -e ssh whawty-auth-master::store /var/lib/whawty
```

On the first connection you will get asked to accept the ssh fingreprint of the master. If you run the command a second time no errors shoulb be shown. After that you can enable the synchronisation by copying the files 'whawty-auth-sync.service' and 'whawty-auth-sync.timer' to '/etc/systemd/system' and enabling the timer using the following commands:

```
# systemctl daemon-reload
# systemctl enable whawty-auth-sync.timer
# systemctl start whawty-auth-sync.timer
```

If you also want to have automatic context-id upgrades on succesfull logins you need to configure the slave to do remote upgrades using the the following as an argument to the do-upgrades command line option:

```
https://whawty-auth-master.example.com/api/update
```

Add a new context id to the store

In order to create a new context id for the store backend you have to generate a new context. This can be done using the script 'gen-auth-context.sh'. You have to specify a context id and a pwcost parameter for this context. The script will print a new conext to STDOUT. Add this line to the auth-store.json config. At the master we will keep the default context for now. Restart the whawty.auth app. Now add the new context to all the slaves store configurations as well. For the slaves you may updated the default context right away as they will not change anything at their store directly. You need to restart the whawty.auth app on the slaves as well. When all slaves are updated you can switch the default context at the master to the new context and again restart the app. After this users logging in on any app (master or slave) should lead to new upgraded password hashes. If all users are upgraded to a the new context you may delete the old.