Vaultwarden

Comment compiler et installer proprement le clone de Bitwarden en Rust. Les bases de données disponibles à ce jour sont PostgreSQL, MySQL et SQLite.

NB: Vaultwarden s'appelait précédemment Bitwarden_rs et a été renommé le 27 avril 2021. Cette page a été remaniée en conséquence. N'hésitez pas à me signaler des soucis (voir l'adresse mail sur ma page de présentation).

NB: J'ai supprimé les informations pour compiler soit-même l'interface web, il fallait NodeJs 14 minimum, ce qui implique de rajouter les explications pour installer cette version de NodeJs. Bref, flemme. Le tutoriel utilise maintenant les *releases* créées par Dani Garcia (auteur de Vaultwarden).

Compilation

Installation des dépendances :

```
sudo apt install pkg-config libssl-dev build-essential
```

Si vous voulez utiliser MySQL comme base de données :

```
sudo apt install default-libmysqlclient-dev
```

Si vous voulez utiliser PostgreSQL comme base de données :

```
sudo apt install libpq-dev
```

Installation de Rust

Installation de rustup, qui nous fournira le compilateur Rust :

```
curl https://sh.rustup.rs -sSf > rustup.sh
```

On n'exécute pas direct un script tiré du web ! On regarde d'abord s'il ne va pas faire de saloperies :

```
vi rustup.sh
```

On le rend exécutable :

```
chmod +x rustup.sh
```

On installe le compilateur Rust (il sera dans notre `$HOME`) :

```
./rustup.sh --default-host x86_64-unknown-linux-gnu --default-toolchain stable
```

> Attention ! Ceci n'est valable que pour l'architecture x86_64 ! Si vous voulez installer Vaultwarden sur une architecture ARM (comme les Raspberry Pi), il faut adapter la commande, a priori en remplaçant `x86_64-unknown-linux-gnu` par `armv7-unknown-linux-gnueabihf`.

On source un fichier qui nous permet de l'appeler

```
source $HOME/.cargo/env
```

Mise à jour de Rust (si vous l'avez déjà installé via rustup)

```
rustup update
```

Compilation

Clonez le projet vaultwarden.

```
git clone https://github.com/dani-garcia/vaultwarden
```

Compilation de Vaultwarden :

```
cd vaultwarden
# Pour les mises à jour
git fetch
git checkout -b "v$(git tag --sort=v:refname | tail -n1)" "$(git tag --sort=v:refname | tail -n1)"
# on supprime les artefacts de la compilation précédente
```

```
cargo clean
cargo build --release --features postgresql
# ou sqlite, ou mysql, selon la bdd que vous souhaitez utiliser
cd -
```

Le résultat de la compilation est dans `vaultwarden/target/release/`.

Récupération de l'interface web

D'abord, si vous ne l'avez pas déjà fait, récupérez la clé GPG de Dani Garcia et de Black Dex, un contributeur :

```
gpg --keyserver keyserver.ubuntu.com --recv-keys B9B7A108373276BF3C0406F9FC8A7D14C3CD543A \
        3C5BBC173D81186CFFDE72A958C80A2AA6C765E1 \
        13BB3A34C9E380258CE43D595CB150B31F6426BC
```

On va utiliser une variable pour le numéro de version, qu'on va aller chercher sur https://github.com/dani-garcia/bw_web_builds/releases, c'est plus simple pour les mises à jour (on change la variables, mais pas les commandes) :

```
VERSION=v2026.4.1
```

Ensuite, on récupère la version patchée de l'interface web avec `wget` :

```
wget -q --show-progress \
      "https://github.com/dani-
garcia/bw_web_builds/releases/download/$VERSION/bw_web_$VERSION.tar.gz" \
      "https://github.com/dani-
garcia/bw_web_builds/releases/download/$VERSION/bw_web_$VERSION.tar.gz.asc"
```

On vérifie la signature de l'archive:

```
gpg --verify "bw_web_$VERSION.tar.gz.asc" "bw_web_$VERSION.tar.gz"
```

On décompresse l'archive :

```
tar xvf "bw_web_$VERSION.tar.gz"
```

Si vous faites une mise à jour, supprimez l'ancienne version de l'interface web :

```
rm -rf vaultwarden/target/release/web-vault/
```

Et on déplace l'interface web dans le dossier où attend le résultat de la compilation de vaultwarden :

```
mv web-vault/ vaultwarden/target/release/web-vault/
```

Si vous faites uniquement une mise à jour de l'interface :

```
sudo rm -rf /opt/vaultwarden/web-vault/ &&
  sudo rsync -a --info=progress2 vaultwarden/target/release/web-vault/  /opt/vaultwarden/web-vault/ &&
  sudo chown -R www-data: /opt/vaultwarden/web-vault/
```

Pour une mise à jour

Suivez le tuto d'installation avec ces précautions préalables :

- coupez le service vaultwarden ;
- faites des sauvegardes de votre installation (fichiers, données de la base de données) avant de faire le `rsync` d'installation (voir plus bas). Pour les fichiers :

  ```
  sudo rsync -a --info=progress2 /opt/vaultwarden/ /opt/vaultwarden_$(date +%F).bak/
  ```

Installation

On va installer Vaultwarden dans `/opt/vaultwarden` et on le fera tourner avec l'utilisateur `www-data` :

```
sudo rm -rf /opt/vaultwarden/deps/* /opt/vaultwarden/build/*
sudo rsync -a --info=progress2 vaultwarden/target/release/ /opt/vaultwarden/
sudo chown -R www-data: /opt/vaultwarden
```

Puis on va créer un service `systemd`, `/etc/systemd/system/vaultwarden.service` :

```
[Unit]
Description=Vaultwarden Server (Rust Edition)
Documentation=https://github.com/dani-garcia/vaultwarden
After=network.target
```

```
[Service]
# The user/group vaultwarden is run under. the working directory (see below) should allow write and read access to this user/group
User=www-data
Group=www-data
# The location of the .env file for configuration
EnvironmentFile=/etc/vaultwarden.env
# The location of the compiled binary
ExecStart=/opt/vaultwarden/vaultwarden
# Set reasonable connection and process limits
LimitNOFILE=1048576
LimitNPROC=64
# Isolate vaultwarden from the rest of the system
PrivateTmp=true
PrivateDevices=true
ProtectHome=true
ProtectSystem=strict
# Only allow writes to the following directory and set it to the working directory (user and password data are stored here)
WorkingDirectory=/opt/vaultwarden/
ReadWriteDirectories=/opt/vaultwarden/

[Install]
WantedBy=multi-user.target
```

Pour l'interface d'administration, on va créer un token avec :

```
/opt/vaultwarden/vaultwarden hash
```

La configuration se fait via des variables d'environnement qu'on va mettre dans `/etc/vaultwarden.env` :

```
SIGNUPS_ALLOWED=false
WEBSOCKET_ENABLED=true
ADMIN_TOKEN=Un token généré avec `/opt/vaultwarden/vaultwarden hash`
ROCKET_ADDRESS=127.0.0.1
WEBSOCKET_ADDRESS=127.0.0.1
SMTP_HOST=127.0.0.1
SMTP_FROM=vaultwarden@example.org
SMTP_PORT=25
```

```
SMTP_SSL=false
```

Vous remarquerez que je dis à Vaultwarden d'envoyer les mails via le serveur SMTP local. À vous de faire en sorte qu'il fonctionne. Allez voir le wiki du projet ou le modèle de fichier d'environnement pour voir quelles variables vous pourriez ajouter, enlever, modifier… Vous pouvez faire ça pour voir les nouveautés :

```
sudo vimdiff -c 'map <F2> :diffget<cr>]czz | map <F3> ]czz | syn off | windo set wrap | winc
h' \
  /etc/vaultwarden.env vaultwarden/.env.template
```

Puis :

```
sudo systemctl daemon-reload
sudo systemctl enable --now vaultwarden
sudo systemctl status vaultwarden
```

Nginx

On installe Nginx s'il n'est pas déjà installé :

```
sudo apt install nginx
```

Configuration du virtualhost :

```
# The `upstream` directives ensure that you have a http/1.1 connection
# This enables the keepalive option and better performance
#
# Define the server IP and ports here.
upstream vaultwarden-default {
  zone vaultwarden-default 64k;
  server 127.0.0.1:8080;
  keepalive 2;
}

# Needed to support websocket connections
# See: https://nginx.org/en/docs/http/websocket.html
# Instead of "close" as stated in the above link we send an empty value.
# Else all keepalive connections will not work.
```

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      "";
}

server {
    listen 80;
    listen [::]:80;
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name vaultwarden.example.org;

    access_log /var/log/nginx/vaultwarden.access.log;
    error_log /var/log/nginx/vaultwarden.error.log;

    ssl_certificate      /etc/letsencrypt/live/vaultwarden.example.org/fullchain.pem;
    ssl_certificate_key  /etc/letsencrypt/live/vaultwarden.example.org/privkey.pem;

    ssl_session_timeout 5m;
    ssl_session_cache shared:SSL:5m;

    ssl_prefer_server_ciphers On;
    ssl_protocols TLSv1.2;
    ssl_ciphers
'EECDH+aRSA+AESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH:+CAMELLIA256:+AES256:+CAMELLIA128
:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD5:!EXP:!PSK:!DSS:!RC4:!SEED:!ECDSA';

    ssl_dhparam /etc/ssl/private/dhparam4096.pem;
    add_header Strict-Transport-Security max-age=15768000; # six months
    gzip off;

    # Redirect HTTP to HTTPS
    if ($https != 'on') {
        rewrite ^/(.*)$ https://vaultwarden.example.org/$1 permanent;
    }

    root /var/www/html;

    # Allow large attachments
    client_max_body_size 525M;
```

```
    location ^~ '/.well-known/acme-challenge' {
        default_type "text/plain";
        root /var/www/certbot;
    }

    location / {
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;

        include /etc/nginx/proxy_params;
        ## /etc/nginx/proxy_params contient normalement ceci :
        #proxy_set_header Host $http_host;
        #proxy_set_header X-Real-IP $remote_addr;
        #proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        #proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://vaultwarden-default;
    }
}
```

Pour créer `/etc/ssl/private/dhparam4096.pem` :

```
sudo openssl dhparam -out /etc/ssl/private/dhparam4096.pem 4096
```

Pour le certificat Let’s Encrypt, on commente le brol relatif à `ssl` puis :

```
sudo nginx -t && sudo nginx -s reload
sudo apt install certbot
sudo mkdir /var/www/certbot/
certbot certonly --rsa-key-size 4096 --webroot -w /var/www/certbot/ --agree-tos --text --
renew-hook "/usr/sbin/nginx -s reload" -d vaultwarden.example.org
```

Une fois qu’on a le certificat, on décommente le brol `ssl` puis :

```
sudo nginx -t && sudo nginx -s reload
```

Sauvegarde

Créer le script de sauvegarde `/opt/backup_vaultwarden.sh` :

```
#!/bin/bash
function vwbackup {
    DATE=$(date '+%a%H')

    # Database, ONLY FOR SQLITE!
    if [[ ! -d /opt/backup_vaultwarden/sqlite-backup/ ]]; then
        mkdir -p /opt/backup_vaultwarden/sqlite-backup/
    fi
    echo ".backup /opt/backup_vaultwarden/sqlite-backup/db.${DATE}.sqlite3" | sqlite3
/opt/vaultwarden/data/db.sqlite3 2>> /opt/backup_vaultwarden/backup.log
    if [[ "$?" -ne "0" ]]; then
        echo "Something went wrong with Vaultwarden database backup, please see
/opt/backup_vaultwarden/backup.log on "$(hostname) | mail -s "Vaultwarden database backup"
youraddress@mail.example.org
        vwbackup
    fi

    # Files
    if [[ ! -d /opt/backup_vaultwarden/files-backup/ ]]; then
        mkdir -p /opt/backup_vaultwarden/files-backup/
    fi
    rsync -a --delete --exclude db.sqlite3 /opt/vaultwarden/data/
/opt/backup_vaultwarden/files-backup/$DATE/ 2>> /opt/backup_vaultwarden/backup.log
    if [[ "$?" -ne "0" ]]; then
        echo "Something went wrong with Vaultwarden files backup, please see
/opt/backup_vaultwarden/backup.log on "$(hostname) | mail -s "Vaultwarden files backup"
youraddress@mail.example.org
        vwbackup
    fi
}
vwbackup
```

Puis :

```
sudo chmod +x /opt/backup_vaultwarden.sh
sudo mkdir /opt/backup_vaultwarden
sudo chown www-data: /opt/backup_vaultwarden
sudo apt install sqlite3 ## Si vous utilisez SQLite
```

Puis, dans le cron de l'utilisateur `www-data` :

```
42 4 * * * /opt/backup_vaultwarden.sh
```

Logs

J'aime bien avoir mes logs dans un dossier dédié pour ce genre de service.

Dans `/etc/rsyslog.d/vaultwarden.conf` :

```
if $programname == 'vaultwarden' then /var/log/vaultwarden/vaultwarden.log
if $programname == 'vaultwarden' then ~
```

Dans `/etc/logrotate.d/vaultwarden` :

```
/var/log/vaultwarden/vaultwarden.log
{
        rotate 52
        dateext
        weekly
        missingok
        notifempty
        compress
        sharedscripts
        postrotate
                invoke-rc.d rsyslog rotate > /dev/null
        endscript
}
```

Puis :

```
sudo mkdir /var/log/vaultwarden
sudo chown root:adm /var/log/vaultwarden
sudo systemctl restart rsyslog
```bash

## Fail2ban

Un fail2ban qui surveille les logs, ça permet de bloquer les petits malins qui font du
```
```

```
bruteforce

```bash
sudo apt install fail2ban
```

Dans `/etc/fail2ban/filter.d/vaultwarden.conf` :

```
[INCLUDES]
before = common.conf

[Definition]
failregex = ^.*Username or password is incorrect\. Try again\. IP: <HOST>\. Username:.*$
ignoreregex =
```

Dans `/etc/fail2ban/jail.d/vaultwarden.local` :

```
[vaultwarden]
enabled = true
port = 80,443
filter = vaultwarden
action = iptables-allports[name=vaultwarden]
logpath = /var/log/vaultwarden/vaultwarden.log
maxretry = 3
bantime = 14400
findtime = 14400
```

Pour la page d'admin, dans `/etc/fail2ban/filter.d/vaultwarden-admin.conf` :

```
[INCLUDES]
before = common.conf

[Definition]
failregex = ^.*Unauthorized Error: Invalid admin token\. IP: <HOST>.*$
ignoreregex =
```

Dans `/etc/fail2ban/jail.d/vaultwarden-admin.local` :

```
[vaultwarden-admin]
enabled = true
port = 80,443
filter = vaultwarden-admin
```
```

```
action = iptables-allports[name=vaultwarden]
logpath = /var/log/vaultwarden/vaultwarden.log
maxretry = 3
bantime = 14400
findtime = 14400
```

Finalement :

```
sudo systemctl restart fail2ban
```

Conclusion

Voilà, vous devriez avoir un serveur Vaultwarden fonctionnel. Plus qu'à aller sur l'interface web que vous venez de mettre en place ou télécharger les clients et à les utiliser !

Pour importer vos mots de passe de Firefox, il faut passer par une application pour les exporter, puis aller dans les outils de votre client (ou de l'interface web).

Révision #78
Créé 2020-01-23 16:16:25 CET par Luc
Mis à jour 2026-05-04 10:40:10 CEST par Luc