

Bitwarden-rs Jail

My company provides a password manager, so I don't need this. But what they provide is closed source. I may just switch over to bitwarden, but perhaps little by little. For now, I just want to get this working.

Like the password manager I'm provided for work, this is a zero-knowledge setup; i.e., the database is stored in an encrypted state, locked by my complex pass phrase. If someone someone gains control of the jail, or even the host system, the database of records/credentials does them no good. It's still scary to put up a publicly-accessible instance, but I'm doing it anyway. Besides, I'll only use it in limited capacity for now, and perhaps I'll put it behind a VPN (Wireguard?) at some point.

Advanced Prep (nullfs)

(Relocate database outside the jail)

If we ever have a problem with this jail and need to blow it away, it would be nice for the database to live on. We can do this! In fact, this is probably one of several steps that could/should be taken to ensure data not specific to the jail is saved outside the jail. We already did this for the BookStack jail (and the majority of this section is a straight copy). Carrying on:

```
zfs create -o compress=lz4 -o atime=off zroot/data/dbs/bitwarden
```

Bitwarden-rs stores the database in a `/data` directory. The standard install creates the `/data` directory in `/home/bitwardenrs/bitwarden_rs_dist`. Due to the fact that this is a thin jail, the `/home` directory (a few directories deep) where the null mount would go cannot be used; so instead we'll mount the `/data` directory over top of the new data dir we'll create.

We will need to adjust the `.env` file accordingly.

```
bastille console bw_jail
```

```
cd /var && mkdir -p db/data
```

Then `exit` from the `su` and `exit` from the console.

```
bastille stop bw_jail
```

Adjust the jail's `fstab`.

# Device	Mountpoint	FStype	Options	Dump	Pass#
/usr/local/bastille/releases/12.1-RELEASE	/usr/local/bastille/jails/bw_jail/root/.bastille	nullfs	ro	0	0
/usr/local/data/dbs/bitwarden	/usr/local/bastille/jails/bw_jail/root/var/db/data	nullfs	rw,late	0	0

```
bastille start bw_jail
```

Below, you'll need to set ownership or permissions on this `/var/db/data`, otherwise bitwarden-rs can't write to it.

On With It

First of all, let's **not** run `tzsetup` on this jail. That gave me problems with 2FA on another instance. Let's try without it.

Grab initial packages/dependencies.

```
bastille pkg bw_jail install -y sqlite3 nginx git sudo vim-console bash node npm python27-2.7.18
```

Adjust

```
# We've got to do a bit of work inside the jail (it'll be easier there)

bastille console bw_jail

# some npm dependency will need to have python2.7 and will fail with python3

cd /usr/local/bin/

# set the symlink

ln -s /usr/local/bin/python2.7 python
```

```
cd -
```

Set up user.

Add new bitwardenrs user to the jail. Set the user below to: bitwardenrs. Enter every line, no need for other configs, only your password

```
adduser -s bash
```

Adjust priv's and log in:

```
# allow sudo, we will use it later
```

```
visudo
```

```
# ADD
```

```
bitwardenrs ALL=(ALL) ALL
```

```
# change to the new user to build and execute our service
```

```
su bitwardenrs
```

```
cd
```

```
id
```

```
# should look like: uid=1001(bitwardenrs) gid=1001(bitwardenrs) groups=1001(bitwardenrs)
```

One add'l step needed: (maybe... try skipping it)

```
bitwardenrs@bw_jail:~ $ exit
```

```
root@bw_jail:~ # chmod 1777 /tmp
```

```
root@bw_jail:~ # su bitwardenrs
```

Another add'l step, and you can't skip this:

Within jail, as root , `cd /var/db && chown -R bitwardenrs:bitwardenrs data`)

Install rust

```
# install latest rust version, pkg version may be outdated and can't build bitwarden_rs

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh

# include the rust env variables

source $HOME/.cargo/env
```

Time to build.

```
# get back to the users home
cd ..

# checkout the latest bitwarden_rs release

git clone https://github.com/dani-garcia/bitwarden_rs/

cd bitwarden_rs/

git checkout "$(git tag --sort=v:refname | tail -n1)"

# and build it with sqlite support

cargo build --features sqlite --release

cargo install diesel_cli --no-default-features --features sqlite-bundled

cd ..
```

"If you need web-vault, we will build it here." Of course we need the web vault.

```
# WEB-VAULT
```

```
# clone the repository
```

```
git clone https://github.com/bitwarden/web.git web-vault
```

```
cd web-vault
```

```
# switch to the latest tag, is not working here, dani-garcia/bw_web_builds lacks v2.12.0 patch
```

```
# export WEB_VERSION="$(git tag --sort=v:refname | tail -n1)"
```

```
# lets use the last working version
```

```
# ^^ use that `export` command instead of the below
```

```
export WEB_VERSION=v2.14.0
```

```
git checkout ${WEB_VERSION}
```

```
# download and apply the bitwarden_rs patch
```

```
curl https://raw.githubusercontent.com/dani-garcia/bw_web_builds/master/patches/${WEB_VERSION}.patch  
>${WEB_VERSION}.patch
```

```
git apply ${WEB_VERSION}.patch -v
```

"Install dependencies and fix some issues."

```
# there is no native freebsd version from node-sass 4.11, lets bump it to 4.12.0
```

```
cat package.json | sed -e 's/"node-sass": "^4.11.0"/"node-sass": "4.13.0"/' | tee package.json
```

```
# I deleted a ^ and changed to 13, from the original write-up I found
```

```
# download submodules
```

```
npm run sub:init
```

```
# manually install angular/compiler-cli
```

```
npm i @angular/compiler-cli
```

```
# install all the other dependencies
```

```
npm install
```

```
# sweetalert used to fail with the latest angular2, but it's been fixed
```

Finally, Build the web-vault

```
npm run dist
```

A 1G RAM VPS instance will run out of memory. Interestingly, it acted incapable of using the swapfile, though I wonder if I could have forced it to.

I had to resize the droplet to get 2G of RAM, and then `export NODE_OPTIONS=--max_old_space_size=4096` (from within the bash shell). And then it works, right? Right.

"At this point we have every components and will have to put them together"

```
cd
```

```
# copy bitwarden_rs dist
```

```
cp -r ~/bitwarden_rs/target/release bitwarden_rs_dist
```

```
cd bitwarden_rs_dist
```

```
# and copy the web-vault files
```

```
cp -r ../web-vault/build web-vault
```

Config

There are `.env` file settings to change. From bitwardenrs user's home dir:

```
cp bitwarden_rs/.env.template bitwarden_rs_dist/.env
```

Edit accordingly (remember, we chose a different data dir to null-fs mount).

```
## Main data folder
DATA_FOLDER=/var/db/data

{...}

## Domain settings
DOMAIN=https://vault.mydomain.tld
```

Set up nginx.

```
su []# be root

bash

# create nginx.conf

cat << EOF >/usr/local/etc/nginx/nginx.conf
worker_processes auto;

events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 6h;
```

```

#server {
    #listen 80;
    #server_name vault.mydomain.tld;
    #return 301 https://$server_name$request_uri;
#}

server {

    listen 10.101.10.120:80;
    server_name vault.mydomain.tld;

    #ssl_session_cache builtin:1000 shared:SSL:10m;
    #ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    #ssl_ciphers HIGH:!aNULL:!eNULL:!EXPORT:!CAMELLIA:!DES:!MD5:!PSK:!RC4;
    #ssl_prefer_server_ciphers on;

    access_log /var/log/nginx/bitwarden_rs_web_vault.log;

    location / {
        proxy_set_header    Host $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://10.101.10.120:8000;
        proxy_read_timeout 90;

        #proxy_redirect http://10.101.10.120:8000 https://10.101.10.120;
    }
}
EOF

# enable and start nginx

sysrc nginx_enable="YES"

nginx -t# test

```



```
service nginx start
```

That's right. Just port 80. It's behind Caddy, remember?

"Create the bitwardenrs init script"

```
mkdir -p /usr/local/etc/rc.conf.d/

# limit the rocket server only to localhost

echo "ROCKET_ADDRESS=10.101.10.120" >/usr/local/etc/rc.conf.d/bitwardenrs # changed to actual

#

cat <<EOF > /usr/local/etc/rc.d/bitwardenrs
#!/bin/sh

# PROVIDE: bitwardenrs
# REQUIRE: LOGIN DAEMON NETWORKING
# KEYWORD: jail rust

# Enable this script by adding:
# bitwardenrs_enable="YES"
# ... to /etc/rc.conf

. /etc/rc.subr

name="bitwardenrs"
rcvar="bitwardenrs_enable"
bitwardenrs_chdir=/home/bitwardenrs/bitwarden_rs_dist
# This is the tool init launches
command="/usr/sbin/daemon"

pidfile="/var/run/${name}.pid"

# This is the tool daemon launches
task="/bitwarden_rs"
procname="/bin/bash"
```

```
command_args="-u bitwardenrs -p \${pidfile} \${task}"
```

```
load_rc_config $name  
run_rc_command "\$1"  
EOF
```

```
sudo sysrc bitwardenrs_enable="YES"
```

```
sudo chmod +x /usr/local/etc/rc.d/bitwardenrs
```

```
sudo service bitwardenrs start
```

Before going lower, be sure to create a CNAME record to catch `vault.mydomain.tld` . Done?
Let's proceed.

Adjust `pf.conf` to allow connections.

Just kidding! We're not touching PF. We've got caddy. Modify the `Caddyfile` in the `caddy_jail`. Add the following:

```
vault.mydomain.tld {  
  
    gzip  
  
    # The negotiation endpoint is also proxied to Rocket  
    proxy /notifications/hub/negotiate 10.101.10.120:80 {  
        transparent  
    }  
  
    # Notifications redirected to the websockets server  
    proxy /notifications/hub 10.101.10.120:3012 {  
        websocket  
    }  
  
    # Proxy the root directory to Rocket  
    proxy / 10.101.10.120:80 {
```

```
transparent
}
}
```

Or is it `encode gzip`? No, that's v2. Your welcome, future self.

Then reload caddy.

```
bastille service caddy-jail caddy restart
```

After Setup! Clean Up!

First, log onto your beautiful self-hosted, powered-by-rust password manager site, and set up an account with an uncrackable password. Then...

This server is open to others to sign up and use. Go into the `.env` file and shut off new user signups!

```
## Controls if new users can register
SIGNUPS_ALLOWED=false
```

And then restart the service or restart the jail. (If you just restart the service, you may be stuck in the terminal, so I just restart the jail.) When you visit and try to sign up again with a new account, it'll pretend to allow you, and then give you a failure warning.

Also, 2FA

Bitwarden-rs allows you to various methods of 2FA. The simplest and most common is an Authenticator app. Do it right away.

References

Adapted from: https://www.ixsystems.com/community/threads/how-to-build-your-own-bitwarden_rs-jail.81389/

More here:

https://www.reddit.com/r/Bitwarden/comments/dg78bi/building_selfhosted_bitwarden_via_bitwarde

[n_rs/](#)

Also: <https://dennisnotes.com/note/20181112-bitwarden-server/> (Ubuntu, Docker, nginx, script install, backup procedure)

Revision #3

Created 5 August 2020 06:59:59 by scoob

Updated 31 July 2021 04:50:03 by scoob