

Loïc Pefferkorn

- [About](#)
- [All posts](#)
- [Projects](#)



Secure Proxmox VE management interface with two-factor authentication, reverse proxy and Let's encrypt certificate

📅 2020-11-21

🔖 [linux](#), [dedibox](#), [sysadmin](#)

- [Proxmox VE](#)
- [Goal](#)
 - [Steps](#)
 - [Safety reasons?](#)
 - [The setup](#)
- [Enable two-factor authentication](#)
- [Create a Let's Encrypt TLS certificate](#)
 - [Register a Let's Encrypt account](#)
 - [Certificate challenge with Gandi DNS live API](#)
 - [Retrieve your Gandi livedns token](#)
 - [Create a new ACME plugin config](#)
 - [Configure and order your first certificate](#)
- [Reverse proxy setup](#)
 - [Nginx configuration](#)
 - [Restrict access to pveproxy port 8006](#)
 - [pveproxy ACL](#)
 - [PVE firewall](#)
- [Resources](#)

Proxmox VE

[Proxmox Virtual Environment](#) (Proxmox VE) is an open-source server virtualization management platform.

The context is self-hosting services for **personal use**, I have been using it for over 5 years and counting!

Goal

Out of the box the [Proxmox Virtual Environment](#) management web ui is only protected by a login form.

The purpose of this post is to show what can be done to **improve its security** besides the obvious strong credentials.

Note: The safest option would be to not expose this interface on Internet and restrict its access through a vpn or a ssh tunnel, but in case these have been lost (broken laptop?) I prefer this last resort option over rebooting the whole host in rescue mode.

Steps

The plan is to:

- Enable [Two-factor authentication](#) for the default **root** user.
- Use a [Let's encrypt](#) TLS certificate instead of the self-generated one, to get rid of the browser warnings/exception.
 - Handle the Let's encrypt renewal challenges through DNS, fully automated with Gandi Live DNS API
- Hide the management web ui behind a **reverse proxy** to:
 - add another **basic HTTP authentication** layer.
 - change the default 8006 port to 443.
 - do not leak a Proxmox node is running there :)

I'll use the command line as much as possible so it's easier to onboard these steps to a configuration management system like Ansible, though I won't share any Ansible snippets because it's too tied to my whole setup.

i Most of these steps should also be doable through the web ui.

Safety reasons?

One may argue the reverse proxy is just **security by obscurity**, however I don't like to expose various services endpoints over the Internet because:

- All software have bugs, therefore **restricting endpoints through a single one** reduces the attack surface, plus is easier to audit/update/restrict.
- The additional **basic HTTP authentication**, in addition to add another layer of safety **avoid revealing what services are running on the host**: the IPv4 space is continuously scanned, and any known service/version found is at some point stored in some black/white-hat database waiting for the next 0-day/CVE vulnerability to quickly break in your host.

The setup

While writing this post:

- Online.net default install of **Proxmox VE 6.2-15**
- The Proxmox VE web management http endpoint will be referred as <https://pve.yourdomain.invalid:8006> (just to be nice to not use any existing domain here)

Enable two-factor authentication

We will set up the **root** user with **TOTP** *Time-based One-Time password* - [This is a well documented process](#)

Note the order: *first* the **root** user will be assigned an OATH key, *then* TFA will be activated in the associated authentication realm.

The **root** user can't be deleted as per [the documentation](#) so I'll just keep it:

The system's root user can always log in via the Linux PAM realm and is an unconfined administrator. This user cannot be deleted, but attributes can still be changed and system mails will be sent to the email address assigned to this user.

Description of the commands below - [pveum\(1\)](#) can come in handy.

- ssh to **pve.yourdomain.invalid**
- Install **qrencode** to ease the key transfer to your favorite 2FA apps
- Generate an OATH key and assign it to the **root** user
- Check the key has been correctly assigned:

```
$ ssh pve.yourdomain.invalid
$ sudo apt install qrencode

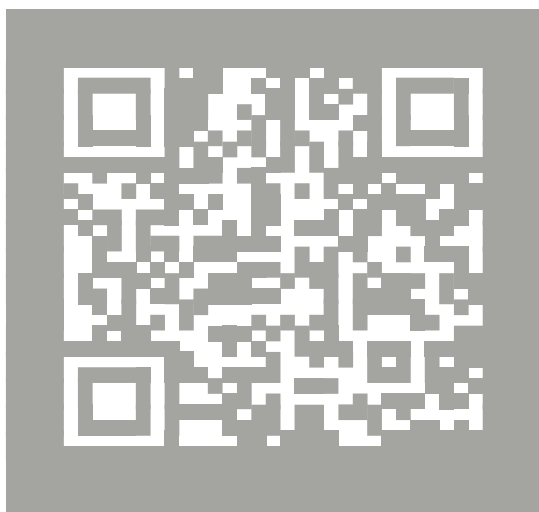
$ KEY=$(oathkeygen) && echo $KEY
RSBWMLBKXKRQQRJR

$ pveum user modify root@pam --keys $KEY
$ pveum user list
```

userid	comment	email	enable	expire	firstname	groups	keys	lastna
root@pam			1	0			RSBWMLBKXKRQQRJR	

- Copy the key to your 2FA apps, I personally use:
 - Unix systems: [pass](#) with the **pass-totp** extension as I've described in [another article](#)
 - Android - [Aegis](#)
- To ease the process a qrcode will be generated:

```
$ qrencode -t ANSIUTF8 -o - $(echo "otpauth://totp/PVE:whatyouwant?secret=${KEY}")
```



- **Keep a ssh session opened on your proxmox host**, just in case something goes wrong with the TFA authentication, to avoid locking yourself out of your host (if that happen, comment out the line **tfa type=oath** under the **pam** section in **/etc/pve/domains.cfg**)
- Enable TFA on the authentication Realm used for the **root** user which is **pam**

```
$ pveum realm modify pam --tfa type=oath
```

```
$ pveum realm list
```

realm	type	comment	tfa
pam	pam	Linux PAM standard authentication	oath
pve	pve	Proxmox VE authentication server	

- Now try to connect to the management web ui at <https://pve.yourdomain.invalid:8006>, you should see the login form reflecting the **Realm** update as shown below (**+ oath**)

Proxmox VE Login

User name:

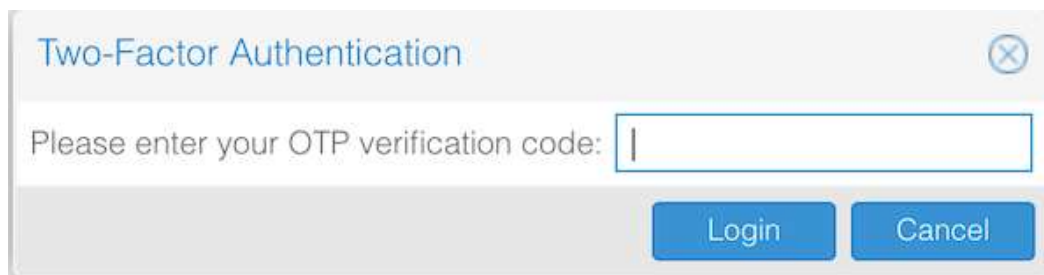
Password:

Realm: Linux PAM standard authentication (+ oath) ▼

Language: English ▼

Save User name:

Right after entering the password you will get the TOTP key prompt:



Two-Factor Authentication

Please enter your OTP verification code:

Login Cancel

And we are done for the tfa part \o/

Create a Let's Encrypt TLS certificate

The benefit of using internal Proxmox tooling to deal with the certificates vs manually installing certbot, is that they will **automatically be renewed by Proxmox**, no need to deal with cron jobs or webserver reload to load the new certificate files as described in [the certificate management section of the PVE admin guide](#):

Automatic renewal of ACME certificates

If a node has been successfully configured with an ACME-provided certificate (either via pvenode or via the GUI), the certificate will be automatically renewed by the pve-daily-update.service. Currently, renewal will be attempted if the certificate has expired already, or will expire in the next 30 days.

Register a Let's Encrypt account

First you need to register a Let's Encrypt account as mentioned in the [documentation](#)

This is achieved with the command **pvenode acme account register default myemail@yourdomain.invalid**

```
$ pvenode acme account register default myemail@yourdomain.invalid
Directory endpoints:
0) Let's Encrypt V2 (https://acme-v02.api.letsencrypt.org/directory)
1) Let's Encrypt V2 Staging (https://acme-staging-v02.api.letsencrypt.org/directory)
2) Custom
Enter selection: 0

Attempting to fetch Terms of Service from 'https://acme-v02.api.letsencrypt.org/directory'..
Terms of Service: https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf
Do you agree to the above terms? [y|N]: y

Attempting to register account with 'https://acme-v02.api.letsencrypt.org/directory'..
Generating ACME account key..
Registering ACME account..
Registration successful, account URL: 'https://acme-v02.api.letsencrypt.org/acme/acct/943674XX'
Task OK
```

Check the **default** Let's Encrypt account has been created:

```
$ pvenode acme account list
default
```

Certificate challenge with Gandi DNS live API

Why the **dns-01** verification challenge and not the **http-01** one?

Because the **http-01** challenge type would require to expose the port 80 to the wild Internet as found in:

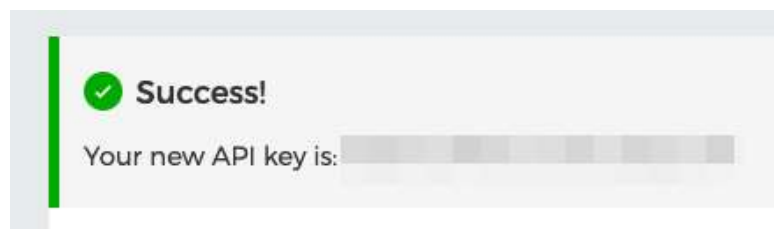
<https://letsencrypt.org/docs/challenge-types/>

```
The HTTP-01 challenge can only be done on port 80.
```

And we don't want that, because it will fail, blocked by the basic http auth that we will add a bit later.

Retrieve your Gandi livedns token

1. Log into your Gandi account
2. On your profile open **User settings** then on **Change password & configure access restrictions**
3. Generate or regenerate your **API key**



Create a new ACME plugin config

1. Write your API key to a file, as this the way it has to be consumed later:

```
$ echo 'GANDI_LIVEDNS_KEY=XXXX' > /tmp/gandi.txt
```

2. Add a new **gandi** ACME plugin configuration fed with that token:

```
$ pvenode acme plugin add dns gandi --api gandi_livedns --data /tmp/gandi.txt
```

⚠ Here you can replace **gandi** with whatever you want like e.g. **vegetables_rock** it won't matter

3. Check it has been registered correctly:

```
$ pvenode acme plugin list
```

```
plugin
```

```
gandi
```

```
<-- yup
```

```
standalone
```

So now we have a valid **gandi** ACME plugin configuration entry setup with our Gandi API key, that could be used to verify the challenge when ordering new certificates.

Configure and order your first certificate

Here we are going to configure and request a certificate for **pve.yourdomain.invalid** , replace with any Gandi domains you own.

1. Add a new configuration with our domain and our previously setup **gandi** plugin config:

```
$ pvenode config set --acmedomain domain=pve.yourdomain.invalid,plugin=gandi
```

2. Finally (🥳) order the certificate - you will be thrilled to notice the validation through the dns challenge with our Gandi api key, provisioning a TXT record:

```
$ pvenode acme cert order
Loading ACME account details
Placing ACME order
Order URL: https://acme-v02.api.letsencrypt.org/acme/order/103073XXX/633361XXXX

Getting authorization details from 'https://acme-v02.api.letsencrypt.org/acme/authz-v3/876572XX'
The validation for pve.yourdomain.invalid is pending!
[Sat Nov 21 17:52:11 CET 2020] Adding record success
Add TXT record: _acme-challenge.pve.yourdomain.invalid
Sleeping 30 seconds to wait for TXT record propagation
Triggering validation
Sleeping for 5 seconds
Status is 'valid', domain 'pve.yourdomain.invalid' OK!
[Sat Nov 21 17:52:58 CET 2020] Removing record success
Remove TXT record: _acme-challenge.pve.yourdomain.invalid

All domains validated!

Creating CSR
Checking order status
Order is ready, finalizing order
valid!

Downloading certificate
Setting pveproxy certificate and key
Restarting pveproxy
Task OK
```

The Let's encrypt certificated has been generated and the management web ui reloaded to take it into account. Reload the page to enjoy your new valid Let's Encrypt certificate!

To get details about the certificates that have been generated (for our next Nginx setup), use the **pvenode cert info** command.

In addition to all the default certificates, you will find the new one toward the bottom:

```
$ pvenode cert info
(...)
```

filename	pveproxy-ssl.pem
fingerprint	10:CA:8A:CF:F6:E3:F3:CD:FA:17:76:D9:7A:DB:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX
subject	/CN=pve.yourdomain.invalid
issuer	/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
notbefore	2020-11-21 17:02:50
notafter	2021-02-19 17:02:50
public-key-type	rsaEncryption

public-key-bits	4096
san	- pve.yourdomain.invalid

The full path of the certificate file and its key are:

- **/etc/pve/local/pveproxy-ssl.pem**
- **/etc/pve/local/pveproxy-ssl.key**

Reverse proxy setup

Nginx configuration

Basically following the instructions from the official wiki -[Web Interface Via Nginx Proxy](#) plus my additional **auth basic twist**:

1. Install the basic version of Nginx and not the full-featured one:

```
$ apt install nginx-light apache2-utils # for htpasswd util
```

2. Overwrite the default config file **/etc/nginx/sites-available/default** with the snippet below, my additional twists are under the comment at the bottom of the **location** block, with are the additional http basic auth and avoid leaking the Nginx version:

```

1  upstream proxmox {
2      server "pve.yourdomain.invalid";
3  }
4
5  server {
6      listen 80 default_server;
7      rewrite ^(.*) https://$host$1 permanent;
8  }
9
10 server {
11     listen 443;
12     server_name _;
13     ssl on;
14     ssl_certificate /etc/pve/local/pveproxy-ssl.pem;
15     ssl_certificate_key /etc/pve/local/pveproxy-ssl.key;
16     proxy_redirect off;
17     location / {
18         proxy_http_version 1.1;
19         proxy_set_header Upgrade $http_upgrade;
20         proxy_set_header Connection "upgrade";
21         proxy_pass https://localhost:8006;
22         proxy_buffering off;
23         client_max_body_size 0;
24         proxy_connect_timeout 3600s;
25         proxy_read_timeout 3600s;
26         proxy_send_timeout 3600s;
27         send_timeout 3600s;
28
29         # Additional twists
30         server_tokens off;
31         auth_basic "hello hello";
32         auth_basic_user_file /etc/nginx/.htpasswd;

```



```
33 }  
34 }
```

3. Generate the `/etc/nginx/.htpasswd` file:

```
$ htpasswd -c /etc/nginx/.htpasswd user1
```

4. Check config and restart Nginx:

```
1 $ nginx -t  
2 $ systemctl restart nginx
```

The Proxmox management web ui is now reachable through <https://pve.yourdomain.invalid> with the Let's Encrypt certificate.

Restrict access to pveproxy port 8006

Last but not least, the port 8006 should only be reachable through Nginx, therefore not exposed on <https://pve.yourdomain.invalid:8006> otherwise the whole reverse proxy trick is pointless.

Unfortunately I did not find any way to configure the interfaces this daemon is listening on :(

What could be done:

- Configure **pveproxy** ACLs to deny anything but traffic coming from 127.0.0.1 (Nginx)
- Use the Proxmox firewall to block port tcp/8006

pveproxy ACL

Configure pveproxy ACL through `/etc/default/pveproxy`, refer to [pveproxy\(8\)](#) for more details:

```
# cat /etc/default/pveproxy  
ALLOW_FROM="127.0.0.1"  
POLICY="deny"
```

This can be tested by starting **pveproxy** in debug mode with `pveproxy start -debug 1` and validate the traffic is correctly accepted/denied when reaching <https://pve.yourdomain.invalid> or <https://pve.yourdomain.invalid:8006>

```
$ pveproxy start --debug 1  
1035: ACCEPT FH10 CONN1  
close connection AnyEvent::Handle=HASH(0x5556a6dd1eaa0)  
1035: CLOSE FH10 CONN0  
1035: ABORT request from x.x.x.x - access denied  
1036: ACCEPT FH10 CONN1
```

PVE firewall

This step is so specific to your setup that there is no one size fits all solution, and there is already a lot of good documentation in the Proxmox documentation or wiki, so I won't duplicate the contents here. (*links below*)

However on my setup this is what I did to only **allow ssh and https traffic to the external ip of my Proxmox host**:

1. **Datacenter>Firewall** level:

- Add/enable 2 new **in ACCEPT** rules for the macros **HTTPS** and **SSH** with your **public pve host IP** as destination

2. **Datacenter>Firewall>Options** level:

- Set **Firewall** to yes

3. **Datacenter>yournode>Firewall>Options** level:

- Set **Firewall** to yes
- Enable firewall logs at the info level: **log_level_in** to info

Quickly check your firewall rules from a distant host, for example with netcat testing if the ports 80, 443, 8006 are reachable or not:

```
$ nc -v -z pve.yourdomain.invalid 8006
nc: connectx to pve.yourdomain.invalid port 8006 (tcp) failed: Operation timed out

$ nc -v -z pve.yourdomain.invalid 80
nc: connectx to pve.yourdomain.invalid port 80 (tcp) failed: Operation timed out

$ nc -v -z pve.yourdomain.invalid 443
Connection to pve.yourdomain.invalid port 443 [tcp/https] succeeded!
```

As desired, only 443 (https) is open.

Now, you can as expected:

- reach your PVE web management interface at <https://pve.yourdomain.invalid> , bound with a **valid SSL certificate**
- have to go through the first **http auth basic authentication** - *hidding the fact a PVE web ui is behind*
- have to go through the **regular Proxmox login**
- and finally have to provide the **two factor authentication token** :)

Hope this post was useful!

Resources

The most up to date Proxmox documentation is [Proxmox VE Documentation Index](#)

- <https://api.gandi.net/docs/livedns/>
- <https://pve.proxmox.com/pve-docs/chapter-pve-firewall.html>
- https://pve.proxmox.com/pve-docs/pve-admin-guide.html#sysadmin_certificate_management
- <https://pve.proxmox.com/wiki/Firewall>

- https://pve.proxmox.com/wiki/Certificate_Management
- https://pve.proxmox.com/wiki/Web_Interface_Via_Nginx_Proxy
- <https://pve.proxmox.com/pve-docs/pvenode.1.html>
- ACME Gandi plugin: **`/usr/share/proxmox-acme/dnsapi/dns_gandi_livedns.sh`**
- PVE local node configuration file: **`/etc/pve/local/config`**
- [pvenode\(1\)](#)
- [pveum\(1\)](#)
- [pveproxy\(8\)](#)
- https://www.reddit.com/r/Proxmox/comments/ed0u7h/question_about_the_pvenode_config_set_command/
- [Nginx - restricting Access with HTTP Basic Authentication](#)

We were unable to load Disqus. If you are a moderator please see our [troubleshooting guide](#).