Securing NGinX Proxy Manager

Securing NGinX Proxy Manager

https://www.youtube.com/embed/UfCkwlPIozw

I've covered NGinX Proxy Manager, a web GUI for NGinX Web Server in multiple videos. I essentially use it anytime I want to give a web site a public URL with Ssl encryption (so basically always).

One of the questions I get fairly often is about how to Secure the admin portal of NGinX Proxy Manager itself. So, in this video I go through the actions with securing the Admin Portal and show you how you can also use the tools in NPM to secure your various self hosted web sites and web applications with even more than just their respective login screens.

Installing NGinX Proxy Manager

Again, I've discussed this, and done this in multiple videos, as well as in multiple posts on this site, but let me cover it again here, so you're not jumping around looking for bits and pieces.

To install NPM you need to install docker and docker-compose, and create a new folder on the server you want to run it in. Next, you'll create two files inside that folder:

- config.json
- docker-compose.yml

Inside the config.json file, you'll put the following:

```
{
  "database": {
    "engine": "mysql",
    "host": "db",
    "name": "npm",
    "user": "<your desired username>",
    "password": "<a strong password>",
    "port": 3306
  }
}
```

And inside the docker-compose.yml file you'll put:

```
version: '3'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    ports:
      - '80:80'
      - '81:81'
      - '443:443'
    volumes:
      - ./config.json:/app/config/production.json
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
  db:
    image: 'jc21/mariadb-aria:10.4'
    environment:
      MYSQL_ROOT_PASSWORD: 'npm'
      MYSQL_DATABASE: 'npm'
      MYSQL_USER: '<username from config.json>'
      MYSQL_PASSWORD: '<strong password from config.json>'
    volumes:
      - ./data/mysql:/var/lib/mysql
```

Make sure to replace the items with < and > around it in each file, and that the username and passwords in each file match.

Now run the command:

`docker-compose up -d`

Give it a minute to pull down everything, and get started, and then in your browser go to the IP address of your server. You should get a Congratulations screen.

if you go tot he IP address at port 81 (http://192.168.1.x:81), you'll be prompted to login to NPM.

Default credentials are:

username: admin@example.com
passwrod: changeme

Make sure to update the email and password, from the default values, then log out, and back in usign the new values you entered.

Now, you're ready to start proxying traffic.

Securing NGinX Proxy Manger Admin Console

The simples and most direct way is to secure NPM to itself. Yep, you just make a loop so that when you ask for a specific URL that you'll have created an A Record for, you get your NGinX Proxy Manager install will proxy the traffic to it's port 81 admin console.

Let's add a new Host entry, and on the dtails page enter the URL you want to use to access the admin console.

In my case, I called it "manage" and created an A Record to point to my public IP address, which is port forwarded 80, 443 to my NPM server.

In my Details tab, I'll enter "manage.example.com" (replace example.com with your domain of course).

Next, enter the IP address of your docker0 interface. You can find this with either:

```
ifconfig
```

or

```
ip addr show docker0
```

Next, enter 81 for the port number.

Turn on the "Block Common Exploits" option, and Save.

We Save now so we can test it and make sure we are routed properly to our Admin login page. This is still unencrypted, so don't log in, but make suer you get to the Admin login page by visiting your URL.

If you get to the page successfully, you can go back to NPM via the IP nad Port, and click the 3 dot icon at the right end of your 'manage' row. Select 'Edit' andn move to the 'SSL' tab.

Choose "Request a New Certificate" from the first dropdown.

Turn on "Force SSL".

Fill in your email address. LetsEnctrypt uses this to let you know if your Certificates have issues or will expire soon and haven't been renewed.

Select to 'Accept the Terms of Service'.

Click 'Save'.

Now, attempt again, to reach your URL. You should be routed to an SSL encrypted site, and you can now login to your Admin console.

What About Keeping the Rest of the Internet Out?

Yep, you just created a publicly accessible URL for your NPM admin console. Not to fear. You can still secure it further. Go to the "Access Lists" tab, and create a new Access List.

Give your new List a name, and then move to the Authorization tab. Enter as many emails and passwords for users you want to have access to the site. If you want to restrict to http basic auth, then save, and close your browser (Note: you may have to clear your cache). Then, re-open, and visit your site, and you should get a prompt for Credentials before you get to the main login screen of NPM itself.

Want more than just basic authentication?

You can also add public IP addresses that will be allowed to access the site. Edit your Access List, and move to the Access tab. Enter yoru public IP address, then try to access the site from a machine not on your LAN, and you'll find you won't be able to.

Now, let's say you want to access the site with one or the other, Username and passsword, or Public IP. Then on the Details tab of your Access List, and enable the option for 'Satisfy Any'. This makes it so either User or IP will be allowed. When this is disabled, then you must have both User credentials and be on an allowed Public IP.

Conclusion

You have amazing open source tools at your fingertips, and making them more secure is highly recommended. Please use the tools and capabilities to run securely.