Website Jail

Before this, I can't think of a time where I edited or wrote html. I can remember creating a basic `index.php` as a test for nginx and/or apache a couple times while tinkering with Nextcloud, but that might be it.

Accordingly, this will be a very basic start of a very simple website. I maybe look forward to doing "cool" complicated stuff in the future, but for now we'll have close to nothing on it. I'm creating the web page because I figure that I might as well have a landing page for the domain itself, but I'm more interested in setting up the reverse proxy work for the subdomains.

To set the expectations properly, the goal is to create an `html` file that renders in a browser by visiting `mydomain.tld`. We'll not be worrying about TLS/https (because `caddy` will eventually do that for us). We'll simply install a web server, create the `html` file, port forward (`rdr`) in `PF` to the jail, and visit in the browser. Someone who's done this a couple times - even if they're documenting it - might be be done in under two minutes. It took me more than two minutes.

Prep

Run the `custom_cshrc.sh` you created in `/usr/local/scripts` to put a custom `.cshrc` file in the jail. Remember, the script just takes the jail name as its only argument.

If desired, adjust the date and time with `tzsetup` or `bastille cmd website_jail tzsetup`.

Web Server

We'll keep it simple and consistent (i.e., BookStack is served by `nginx`), so we'll install `nginx`.

```
bastille pkg website_jail install -y nginx vim-console
```

And then we'll enable it and start it.

```
bastille sysrc website_jail nginx_enable="YES"
```

```
bastille service website_jail nginx start
```

We'll configure it in a moment.

Internet Content

That sure is a fancy title for a bare `html` file.

Let's just hop into the jail console for a few minutes.

```
bastille console website_jail
```

And we'll head to the usual FreeBSD spot, create a website directory, and then file.

```
cd /usr/local/www
```

```
mkdir mydomain.tld && cd mydomain.tld
```

```
vim index.html
```

And we will create our initial homepage.

```
<!DOCTYPE html>
<html>
<body>

<h1>We Did It!</h1>

<p>How exciting.</p>

<p>Be sure to check out all the great related services.  Links coming soon...</p>

</body>
</html>
```

Configuration

Now we can create our configuration in `nginx` so it knows how to listen and what content to serve.

```
vim /usr/local/etc/nginx/nginx.conf
```

In theory, all we have to do is change `server_name localhost` to `server_name mydomain.tld www.mydomain.tld` and change `root /usr/local/www/nginx` to `root /usr/local/www/mydomain.tld`. With any luck, we can reload `nginx` and be ready to test (almost).

Before moving forward, `exit` out of the jail console.

First we test the config (even though the test is built into the reload).

```
bastille cmd website_jail nginx -t
```

If successful, we perform the reload.

```
bastille service website_jail nginx reload
```

Testing It Out

You'll need the jail's IP for this, which you can get from `bastille list`.

Then there needs to be a redirect rule in `PF`, which is basically port forwarding. There's an example already in `/etc/pf.conf`, so it just needs to be uncommented, and updated with the website jail's internal IP.

```
# the macro
website_ip = "10.101.10.140"

# the port forward
rdr pass inet proto tcp from any to any port {80, 443} -> $website_ip
```

And it needs to be tested with:

```
pfctl -vnf /etc/pf.conf
```

Hiccup

NameCheap.com provides a default CNAME record that redirects my internet traffic to their "parking page" and I hadn't deleted that yet, so I had to wait on it to die.

Visiting the IP address does successfully display the webpage, but it would have been nice to see DNS do what it's supposed to too. Of course, it worked via hostname eventually.

Last Step

Remove those rules from `pf` and force reload `pf`. We will be using `https` in no time flat after the next jail is up.

Revision #4
Created 5 August 2020 04:08:02 by scoob
Updated 24 September 2020 06:26:12 by scoob