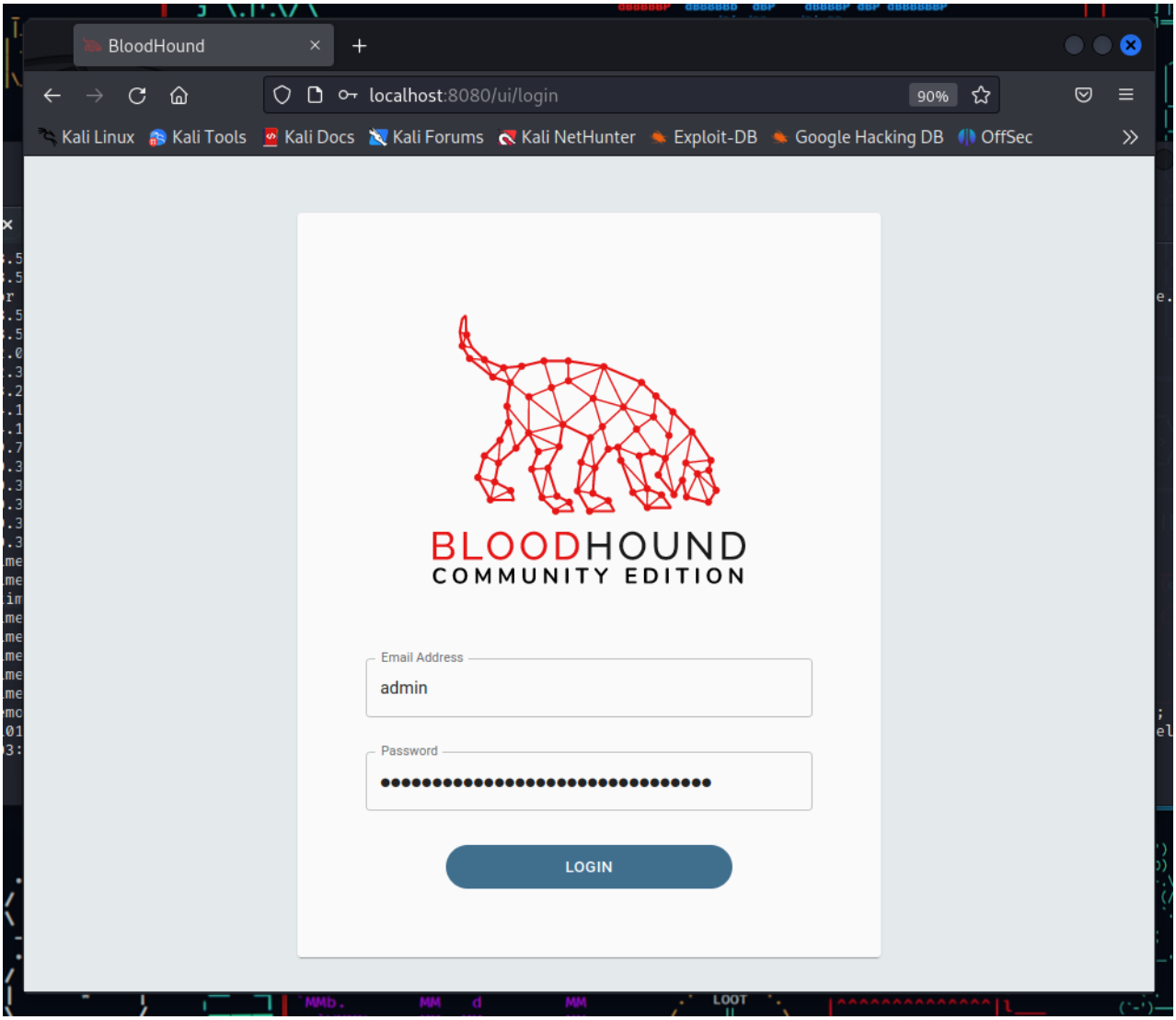
Deploying BloodHound Community Edition for Pentesters

blog.spookysec.net/Deploying-BHCE

August 19, 2023



Hello everyone!

I know this post is a bit late, blame Blackhat/DEFCON, but I wanted to put a guide together for deploying BloodHound Community Edition for Pentesters and Red Teams. This is going to be more of an Administration guide than a “how to use cool new thing” - the core components of BHCE are the same as BloodHound Classic.

So, the reason for writing this post: I deployed BHCE at the company I work for and I ran into some things I’ve never had to do before, so I thought this might be a good topic to write a blog post on. So, here’s all the topics we’re going to cover today:

- Deploying BloodHound Community Edition
- Modifying the BloodHound Config File
- Creating User Accounts

- Setting up Multi-Factor Authentication
- Deploying SSL/TLS
- Backgrounding Docker

Note: *that some sections might be a little bit longer and exact steps may change from version to version. I'll try my best to keep this updated, but I generally don't update older blog posts*

Deploying BloodHound Community Edition

Before we begin, some pre-requisites are required:

- A Linux *or* Windows Server (We'll be using Linux)
- Network Access
- Docker/Docker-Compose

That's pretty much it! As long as you have those things, you should be able to get started. I'll be using Kali for the deployment, we can install Docker with the following command:

```
apt install docker-compose
```

This will install Docker-Compose, Docker, and all it's dependencies. As of 2025, the installation instructions have changed slightly to make it easier to rapidly deploy BloodHound Community. This can now be done with a tool called **BloodHound-CLI**, it can be downloaded here:

```
mkdir /opt/bloodhound && cd /opt/bloodhound && wget
https://github.com/SpecterOps/bloodhound-cli/releases/latest/download/bloodhound-
cli-linux-amd64.tar.gz && tar -xzf ./bloodhound-cli-linux-amd64.tar.gz && rm
bloodhound-cli-linux-amd64.tar.gz
```

Please note that installation instructions for Windows and ARM systems may vary. For updated installation instructions, please refer to the BloodHound Community Edition Quickstart guide by SpecterOps

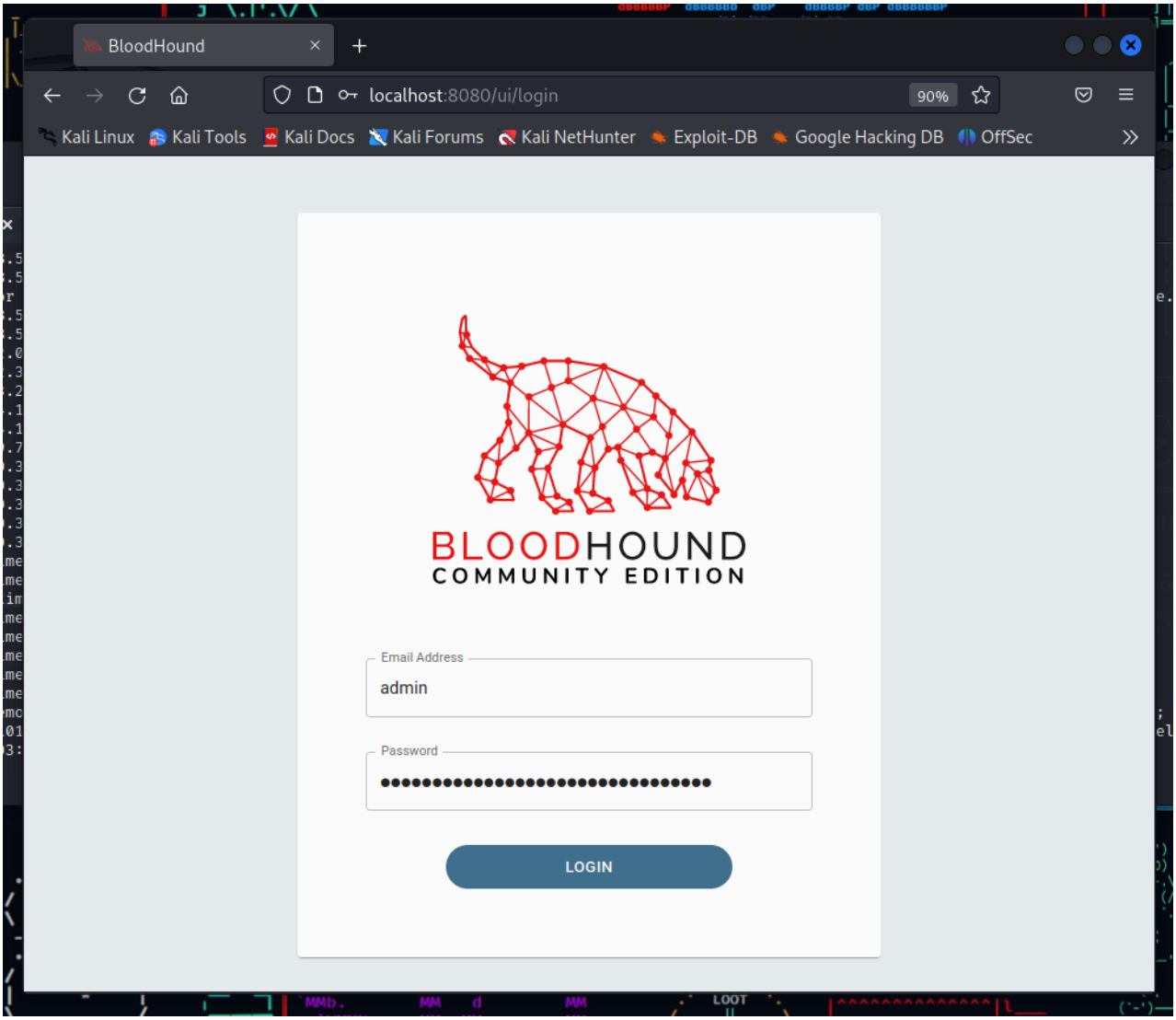
This command will download BH-CLI into the /opt/bloodhound folder. This tool enables you to do all sorts of cool stuff, so if you're not handy with Docker (Like me!), you can easily update BloodHound, start/stop it, display logs, reset the admin password, and more. You can check out the full functionality with the `./bloodhound-cli --help` command.

To get BloodHound up and running, simply run:

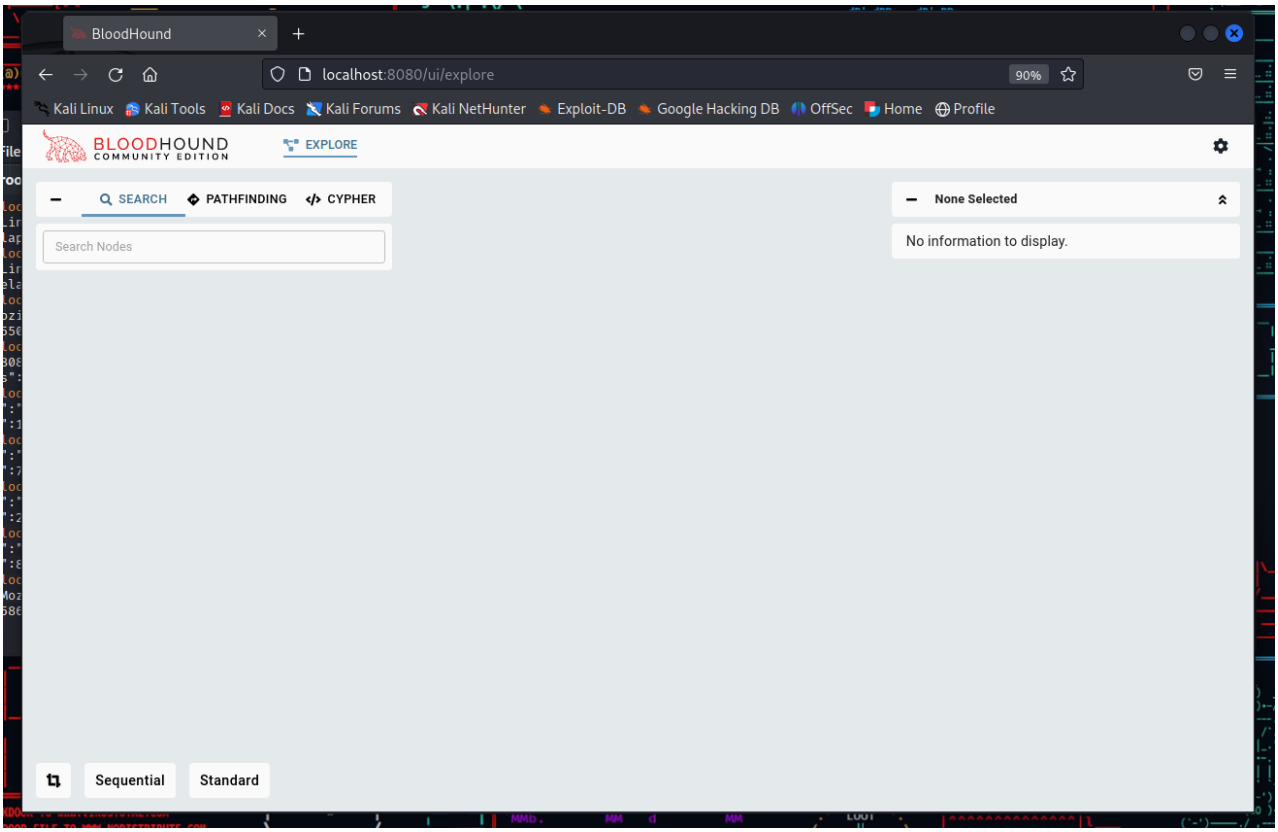
```
./bloodhound-cli install
```

And that's it! You'll be up and running, the admin password will be provided in the terminal output for you to login with. If, for some reason it does not show up for you, you can fetch the password with `./bloodhound-cli config get default_password`

For a single user deployment this will suffice and give us everything we need to get started. It doesn't really scale for multi-user deployments, however. We'll cover that in the Deploying SSL/TLS section though.



Upon logging in, you will be prompted to change the default password. This password must meet a certain set of complexity requirements.



Modifying the BloodHound and Docker Config Files

Something super important to note is this may expose you to potential attacks. Often BloodHound may be used on hostile networks like HackTheBox, TryHackMe, or another environment, so additional hardening steps may be a good idea. I'm going to run with this to show you a **basic** modification of the config files - later we'll deploy SSL/TLS.

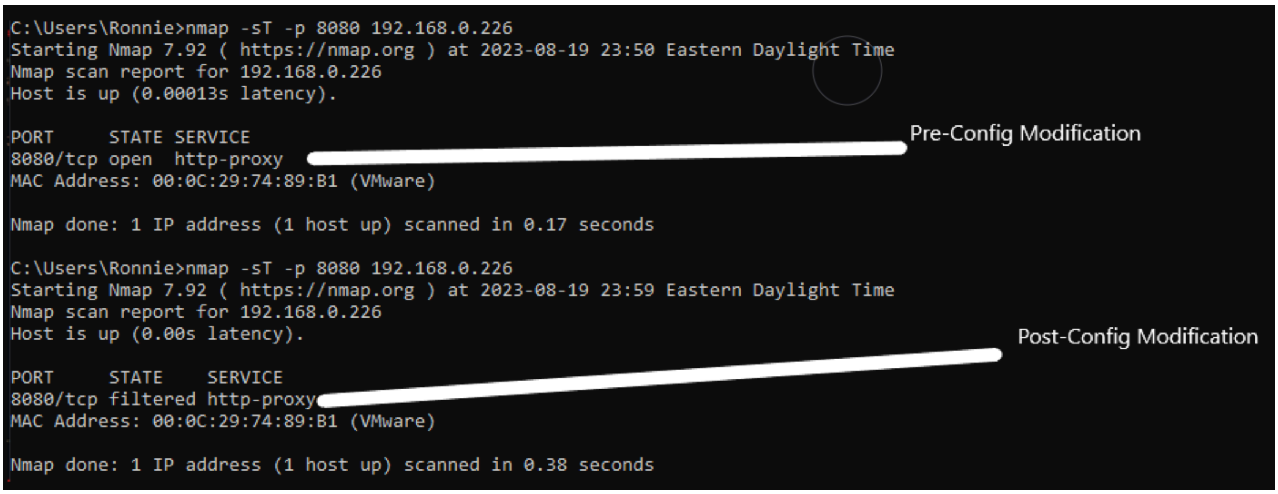
If you are using this for you and you only, it may be worth modifying the config file to bind to **127.0.0.1** instead of **0.0.0.0**. This can be done by modifying the Bloodhound.config.json file in /opt/BloodHound.

```
{"bind_addr":"127.0.0.1:8080","collectors_base_path":"/etc/bloodhound/collectors",
"default_admin":{"password":"...","principal_name":"admin"..."tls":
{"cert_file":"","key_file":""},"version":1,"work_dir":"/opt/bloodhound/work"}
```

We will also have to modify the Docker-Compose.yml file. Note that by default a config file is not specified. We want to uncomment these lines so our config file is used.

```
bloodhound:
  image: docker.io/specterops/bloodhound:${BLOODHOUND_TAG:-latest}
  environment:
    - bhe_disable_cypher_qc=${bhe_disable_cypher_qc:-false}
  ports:
    - ${BLOODHOUND_PORT:-8080}:8080
  ### Uncomment to use your own bloodhound.config.json to configure the application
  # volumes:
  #   - ./bloodhound.config.json:/bloodhound.config.json:ro
  depends_on:
    app-db:
```

After uncommenting the lines, we can restart the Docker container by pressing ctrl+c to kill the process. Afterwards, we can re-execute `./bloodhounc-cli containers up`, and the container will restart.

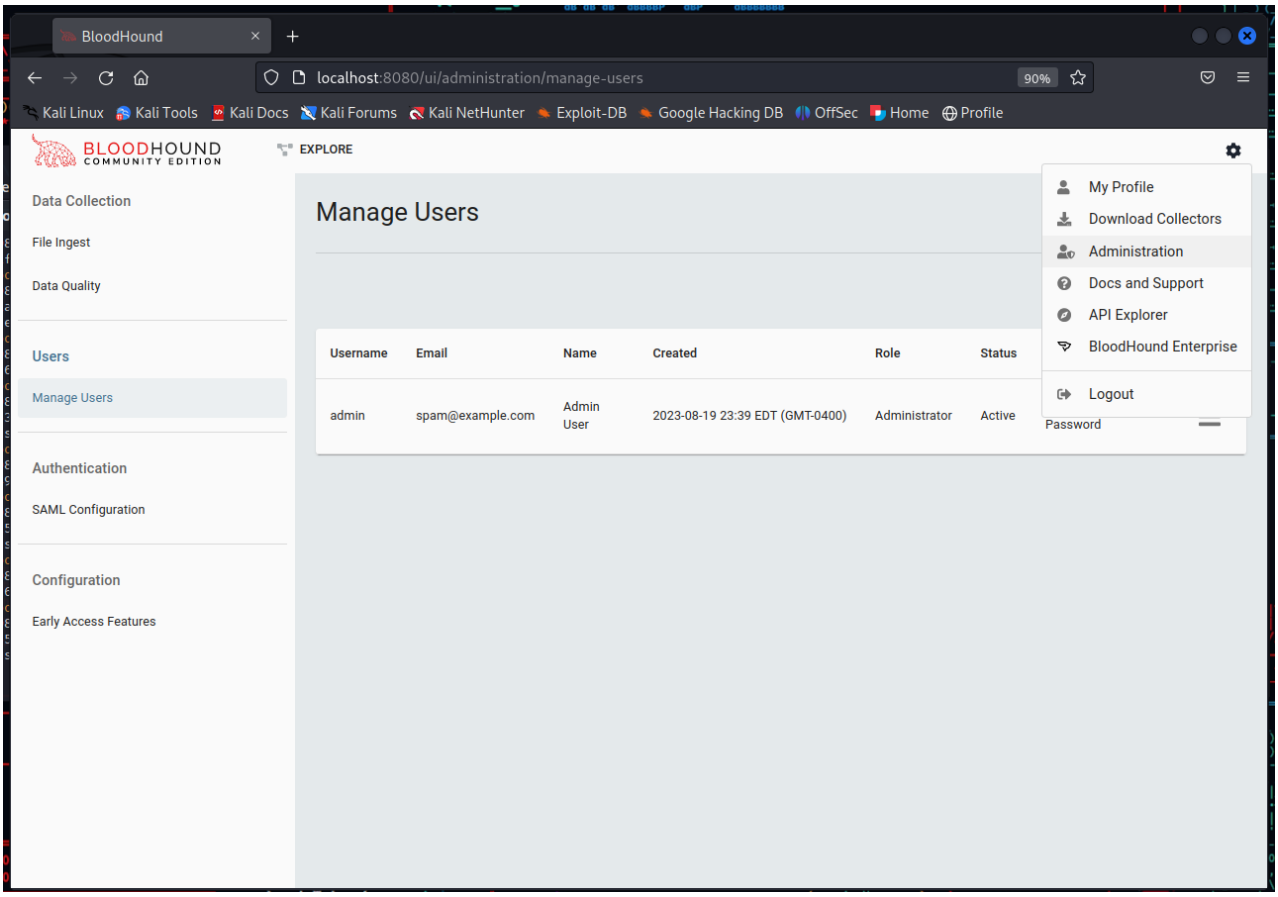


Now, if an attacker tries to connect to our BloodHound instance, they will be unable to do so. Keep this in mind while working within hostile environments! For the blog post, we are going to assume you're working with a **team** and not in an individual setting, so we're going to revert this config back to 0.0.0.0:8080.

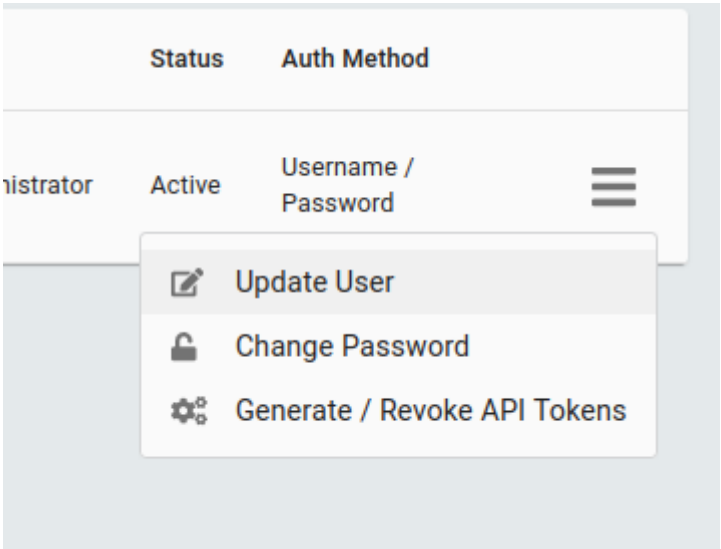
```
{"bind_addr":"0.0.0.0:8080","collectors_base_path":"/etc/bloodhound/collectors","d
efault_admin":{"password":"...","principal_name":"admin"..."tls":
{"cert_file":"","key_file":""},"version":1,"work_dir":"/opt/bloodhound/work"}
```

Creating User Accounts

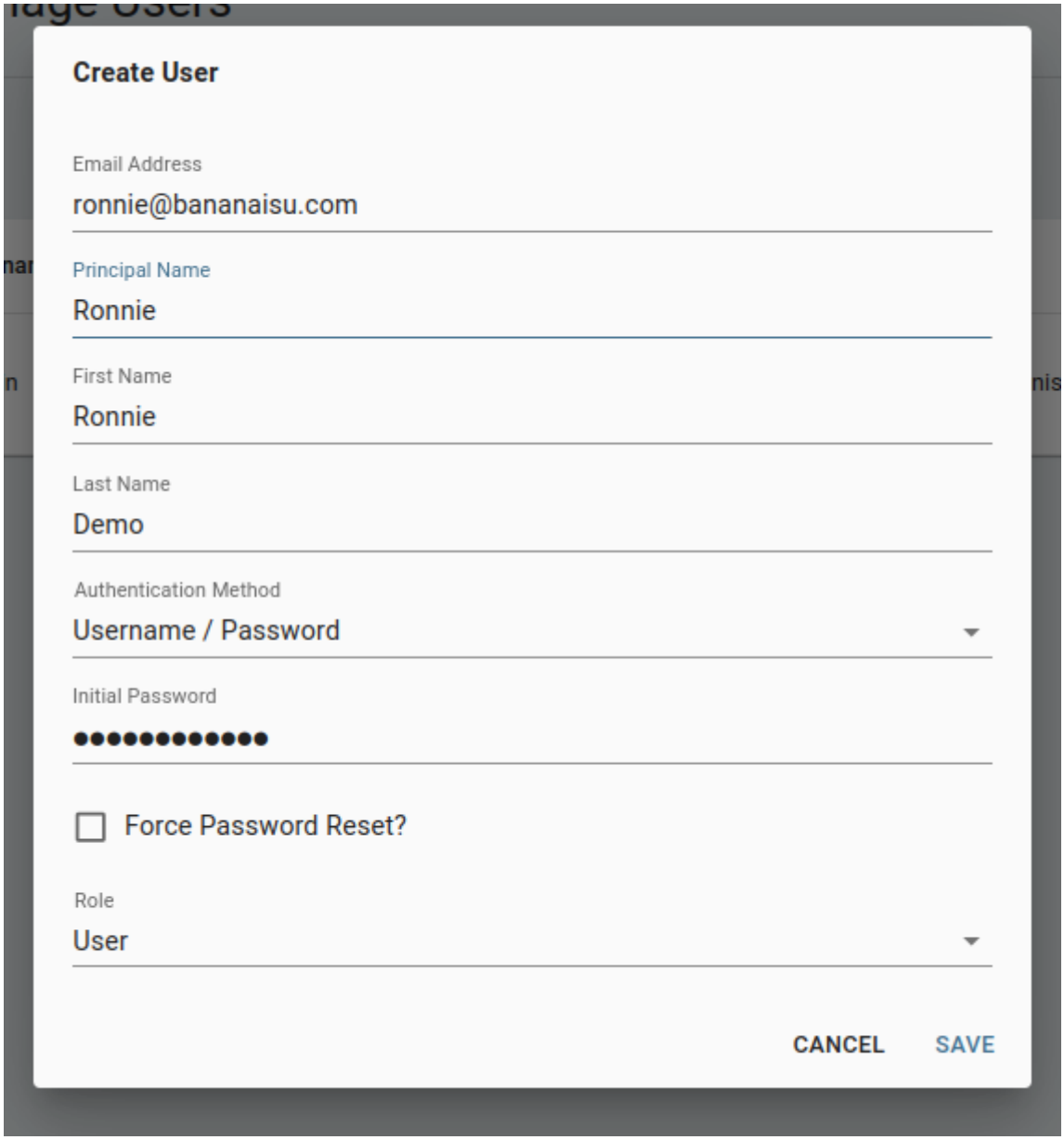
As always, in a team setting it's bad practice to use the default Administrator user as your personal user account, so in this next section, we'll show you how to create a new user. We can do this by navigating to the settings cog, selecting "Administration", then "Manage Users".



By default, there will be one user, Admin with the email spam@example.com. If you want to modify this, you can select the “hamburger” menu and select “Update User”.

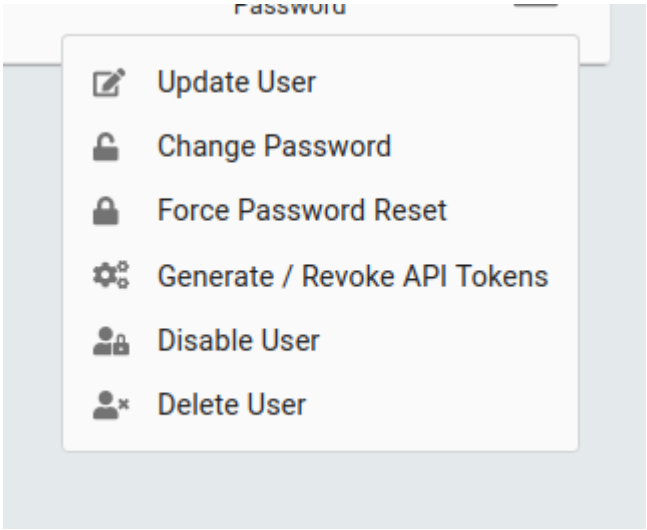


Selecting “Create User” will open a new dialogue box:



In here, we can assign one of four roles, Read Only, Upload Only, User and Administrator. Note that currently the User role does not have permission to upload files. This may be a misconfiguration, so for the time being, I would recommend having a dedicated user account to upload files as Administrator is **very** broad permission wise. For a single user deployment, Administrator should be fine, if you're working in a team, I would opt for a dedicated File Upload role. This process can be repeated for each member of the team.

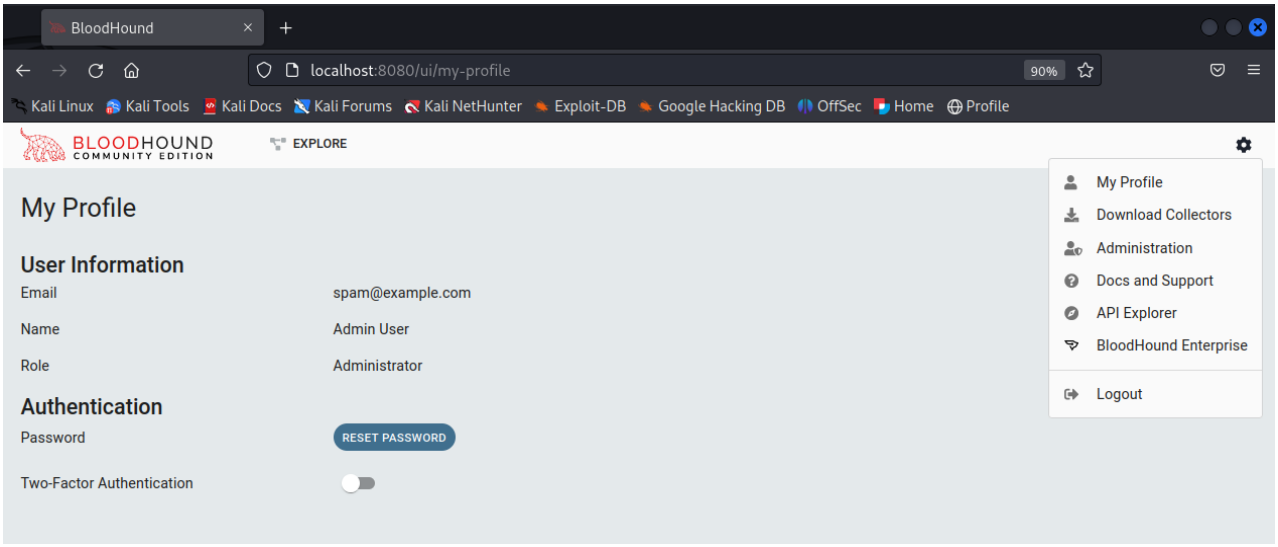
For awareness, it's possible to disable, force password changes, change passwords, generate API tokens, and delete user accounts as well.



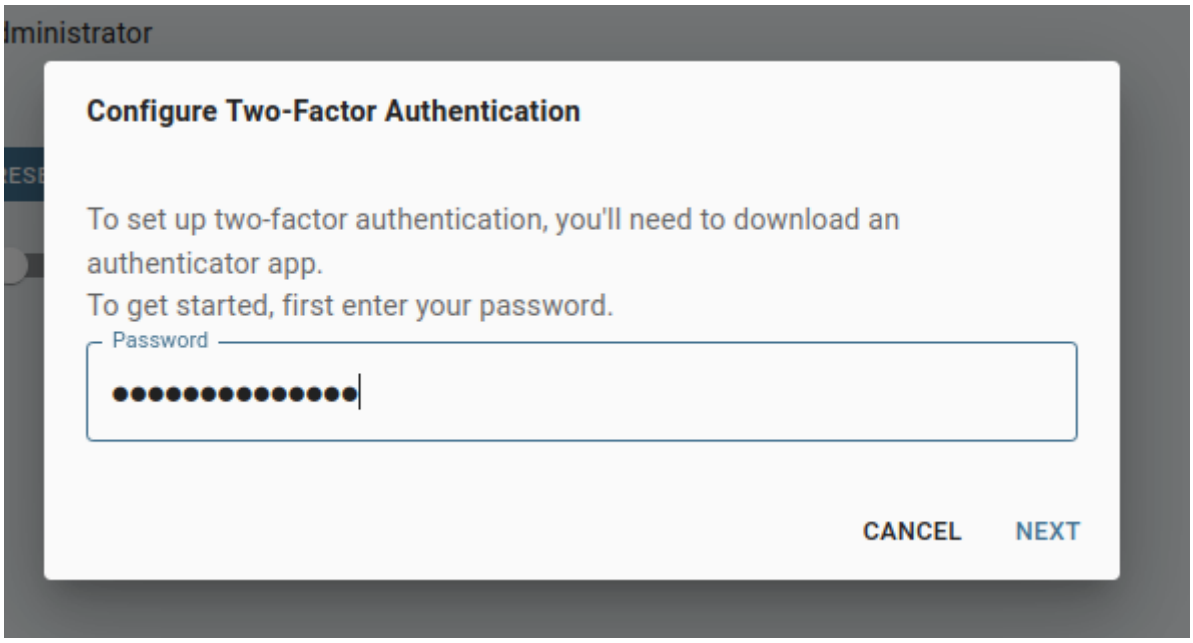
Setting up Multi-Factor Authentication

Continuing our hardening best practices, multi-factor authentication can be setup on per-user basis. Unfortunately, it does not look like there is a way to force MFA at this time, so it would be best to setup an internal policy that requires each user to have MFA configured due to BloodHound containing sensitive data.

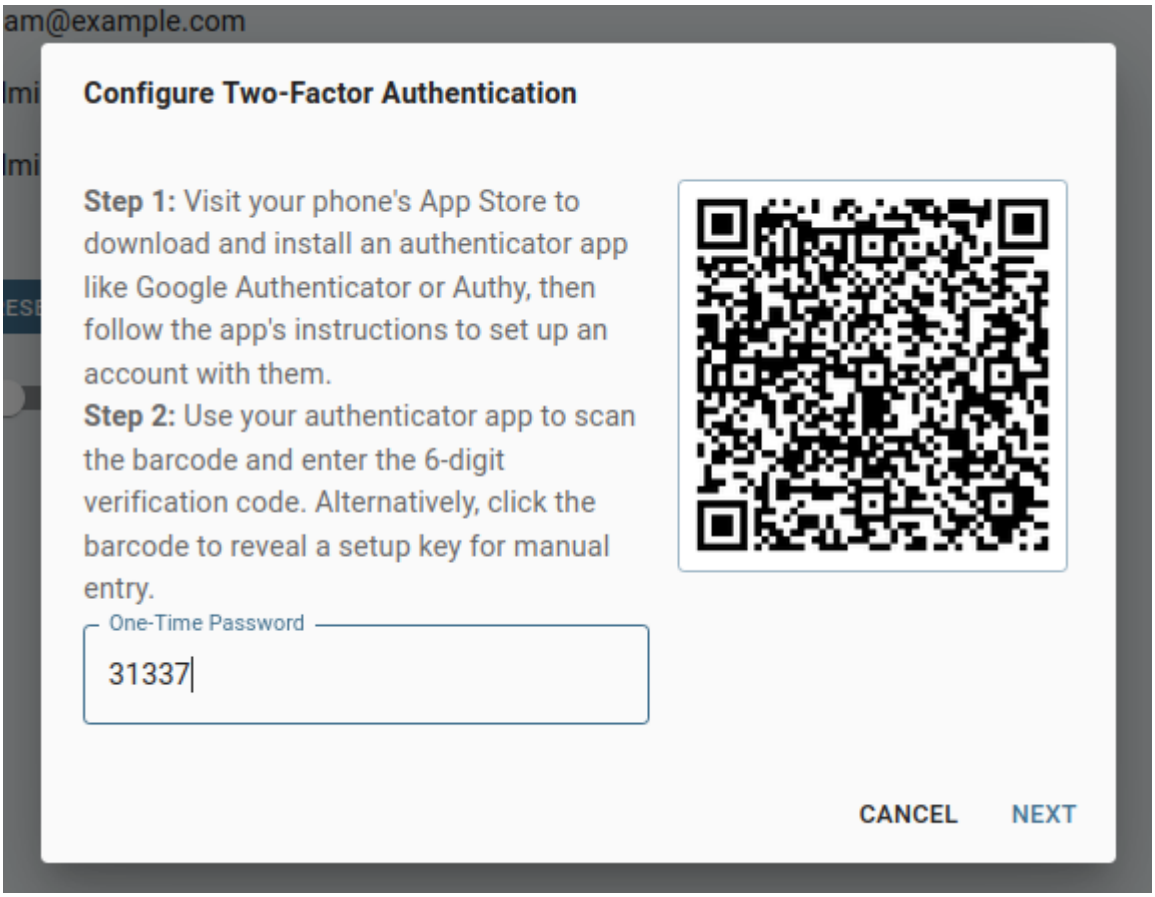
MFA can be configured by navigating to the settings cog and then selecting “My Profile”.



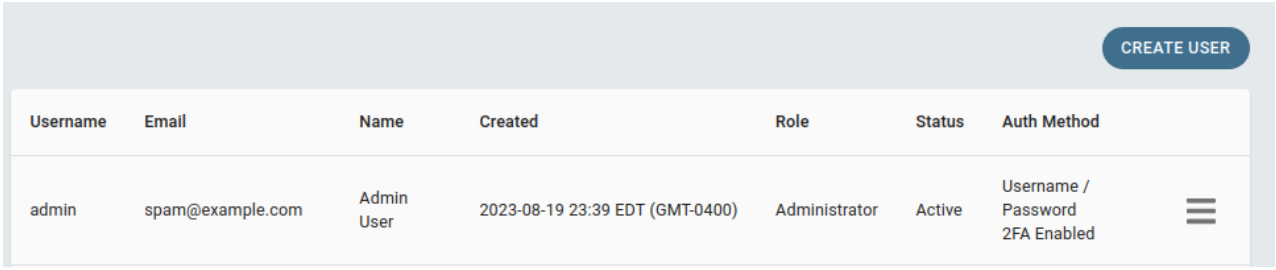
You can then click the toggle button for MFA - You will be prompted to enter your current password.



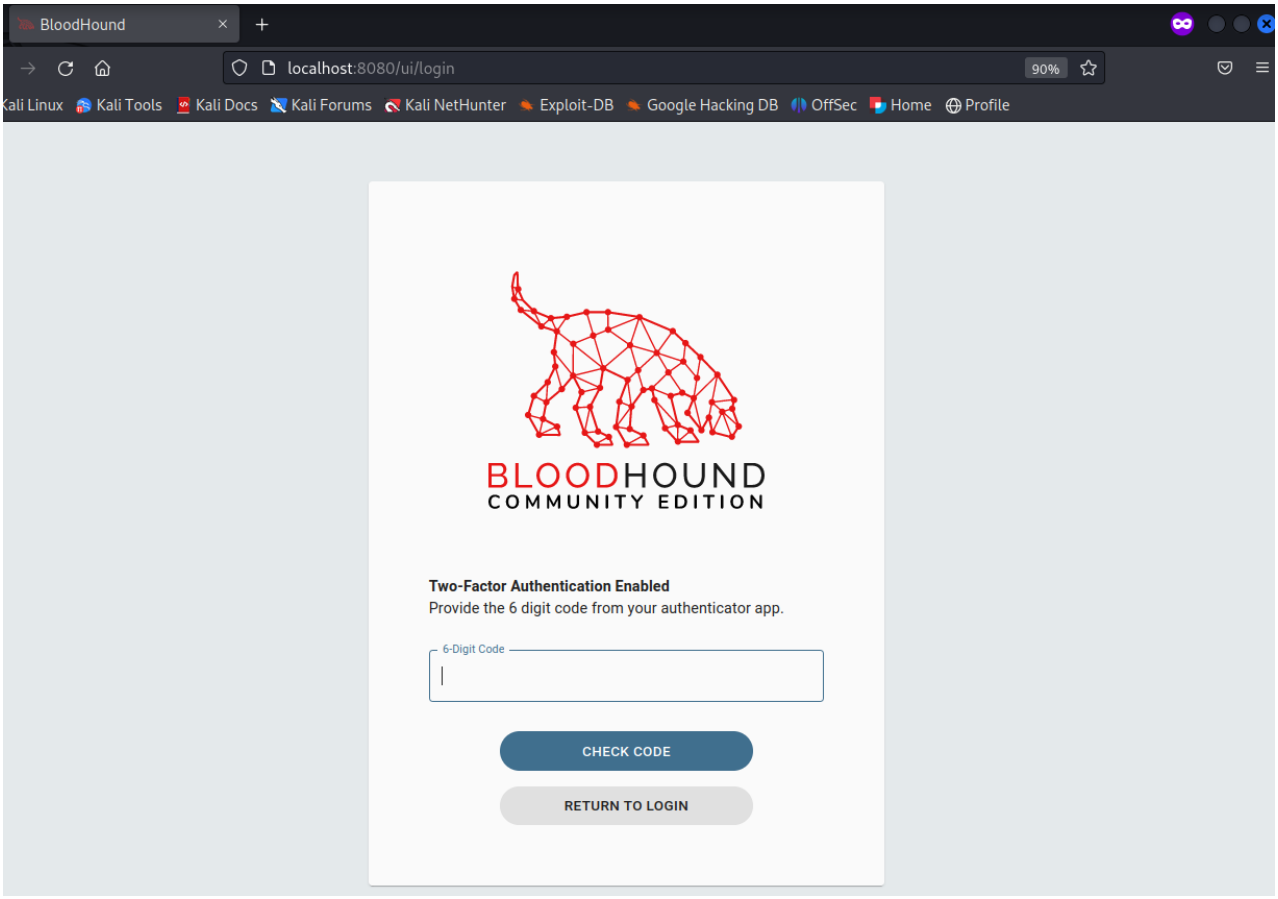
You will then be prompted to scan a QR code for an authenticator app such as Authy, gAuth, RSA SecurID, Microsoft Authenticator, or another app.



If you input the code correctly, you should receive a prompt that says you'll need to use your Password and TOTP code. Navigating back to the "Manage Users" page, you'll see that the user is now flagged as having MFA enabled.



The next time you login, you'll be prompted to input your MFA token.



And that's it, that's the MFA setup. Super simple. If you need to disable MFA, you just need to go back to your profile and select the toggle button. It will require you to input your password again, and then MFA will be disabled. There does not currently appear to be a way to reset the MFA token, so if you lose your token it may be lost for good and you may have to delete/re-create the user account, so be careful!

Deploying SSL/TLS

SSL/TLS is absolutely necessary in my opinion if you're working in a team - by default BHCE does **not** come with a self-signed certificate, it only runs on HTTP as you may have already noticed. This is a bad idea for a number of reasons, most importantly, you don't want someone to snoop on your password! So - how can we deploy HTTPS?

Great question. I struggled with this for quite a bit since I had no prior experience with Docker. Generating SSL/TLS certs are fairly easy, most often this consists of submitting a request to your PKI team. You may have to do some OpenSSL magic to create a .cer and

a .pem file. Make sure that your .pem file is **not** password protected. There are plenty of guides out there on how to extract various things from various formats. I'm going to use CloudFlare to create a SSL certificate for me.

I recommend placing these files in a location other than /opt/BloodHound, though I'm going to put them there for the ease of use. We'll have to modify our config file to include our certs, like so.

```
..."tls":
{"cert_file":"/opt/bloodhound/cert.cer","key_file":"/opt/bloodhound/cert.pem"},...
```

Now, if we try to start the application right now, we'll receive an error that says something like this:

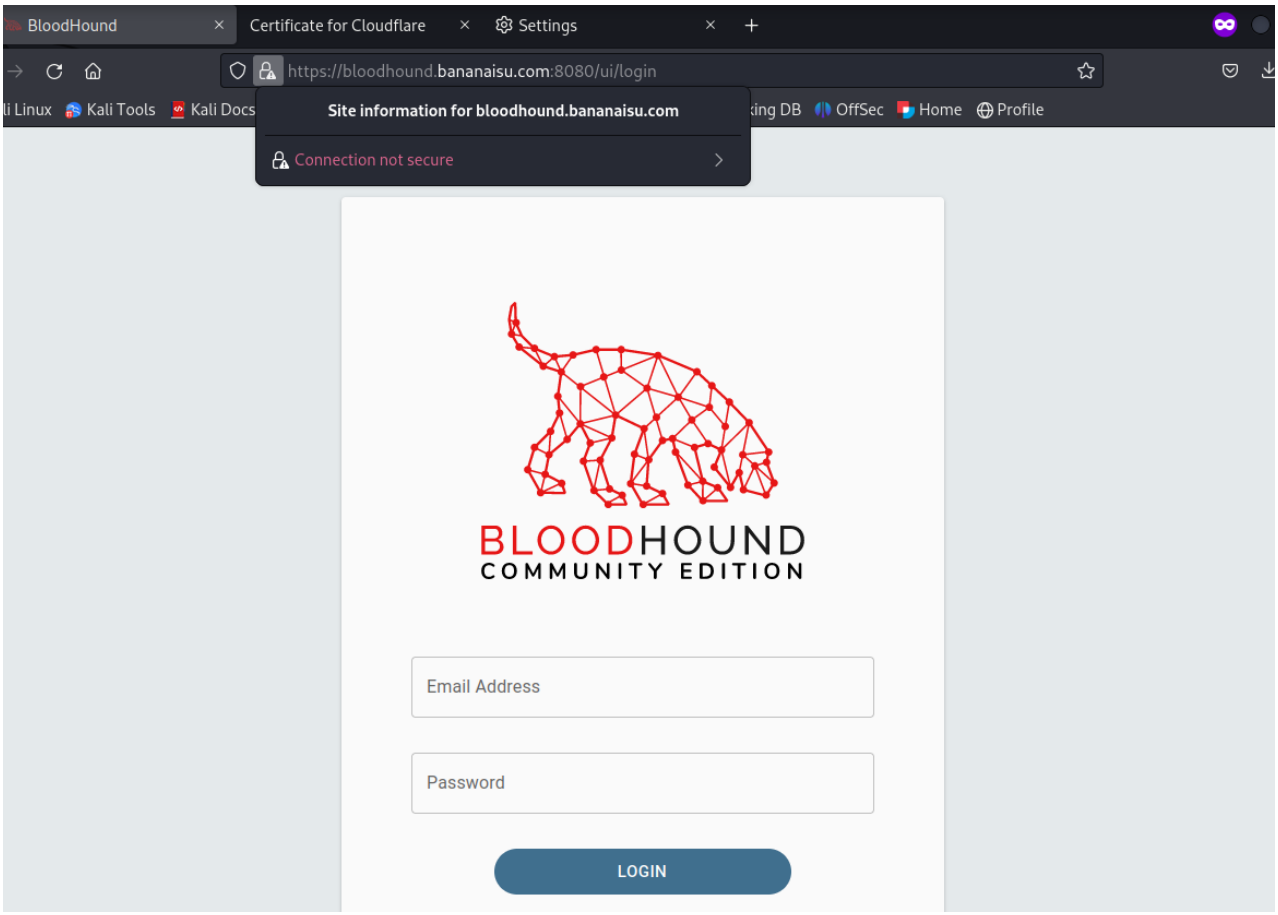
```
bloodhound_1  | {"level":"error","time":"2023-08-
20T04:50:35.429379169Z","message":"HTTP server listen error: open
/opt/bloodhound/cert.cer: no such file or directory"}
bloodhound_1  | {"level":"error","time":"2023-08-
20T04:50:35.429503069Z","message":"HTTP server listen error: open
/opt/bloodhound/cert.cer: no such file or directory"}
```

This is because the files are **not** on the containers filesystem - only on the hosts. So, we must modify the Docker configuration file to share a mount point. We can do this by adding a line right below our specified configuration file;

```
    volumes:
      - ./bloodhound.config.json:/bloodhound.config.json:ro
      - /opt/BloodHound/:/opt/bloodhound/:ro
```

This is going to share **all** the files in /opt/BloodHound to /opt/bloodhound on the docker container. So note that if you have **any** sensitive files in this directory, they can be accessible in this directory, individual files can be shared as well, which may be more preferable. Note in the above line we can specify this as Read Only which makes it *slightly* more secure.

Anyways - security rant concerns aside, after saving this file and putting the certificates in /opt/bloodhound/ and having them named cert.cer/cert.pem (or whatever you would like to name them), we can restart the server. Once again, this can be done by ctrl+c to stop the process and re-running `./bloodhound-cli containers up`. Now, you should see our site is served with an SSL certificate! Our communications between us and the site are now encrypted. Note that I'm using a self-signed SSL cert for our demo here. I'm not trying to make things overly complex, though you should use a signed/trusted certificate in production.



And that's it, we now have HTTPS!

While we're on the topic of best practices, there's a few things that I think are worth mentioning you change/investigate if you're interested in tweaking the config files some more:

- If you'd like to directly connect to the Neo4j database, you have to uncomment two lines in the config file.
    - By default, the username and password for BHCE's Neo4j database is neo4j:bloodhoundcommunityedition
        This should be changed before making them public facing.
    - PostgreSQL suffers the same issue
- Binding to a specific interface's IP address instead of quad 0 may be better if you don't want to bind to your loopback interface.

Updating BHCE

With the BloodHound-CLI change, it's much easier to update BHCE now, by simply running `./bloodhound-cli update` command, BloodHound will attempt to fetch the latest versions of the containers from the docker registry. If for some reason this fails, you can always use the `docker-compose pull` command to update them as well.

Managing the Docker Containers

This is our last section - As of the BloodHound-CLI change, SpecterOps has made it super easy to manage the docker containers using the bloodhound-cli app. To start and stop the containers, we can run the following commands:

```
./bloodhound-cli containers stop #This will stop our containers
./bloodhound-cli containers start #This will start our containers
```

If you need to bring all of the containers down, remove them and rebuild then (sometimes useful for updates), you can use the following:

```
./bloodhound-cli containers down # This will bring down and destroy the containers
./bloodhound-cli containers up # This will rebuild a fresh copy of the containers
```

If something happens where BloodHound-CLI isn't working as expected, manual management of the containers can be done with the Docker-Compose commands as well. I would highly recommend reaching out in the Slack Channel if you come across any issues. No tool is perfect, so if you run into any bugs or have any questions, make sure you reach out - https://ghst.ly/BHSlack

```
docker-compose up -d # This will bring the containers up and daemonize them
docker-compose down # This will bring the containers down
```

I hope this helps :D

~ Ronnie