Deploying BloodHound Community Edition for Pentesters

blog.spookysec.net/Deploying-BHCE

19 Aug 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 Server
- 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. Next, we’ll want to pull down two files from the **SpecterOps BloodHound** repository, it can be found here. Be careful not to confuse it with the BloodHoundAD BloodHound repository. That repository will soon be archived and will no longer be supported. Anyways - continuing on. From the repo, we want to grab the following files:

- Docker-Compose.yml
- BloodHound.config.json

I recommend creating a new folder in `/opt/` called "BloodHound" to store these files, alternatively, you can just clone the whole repository and copy the BloodHound.config.file. This can be done with the following commands:

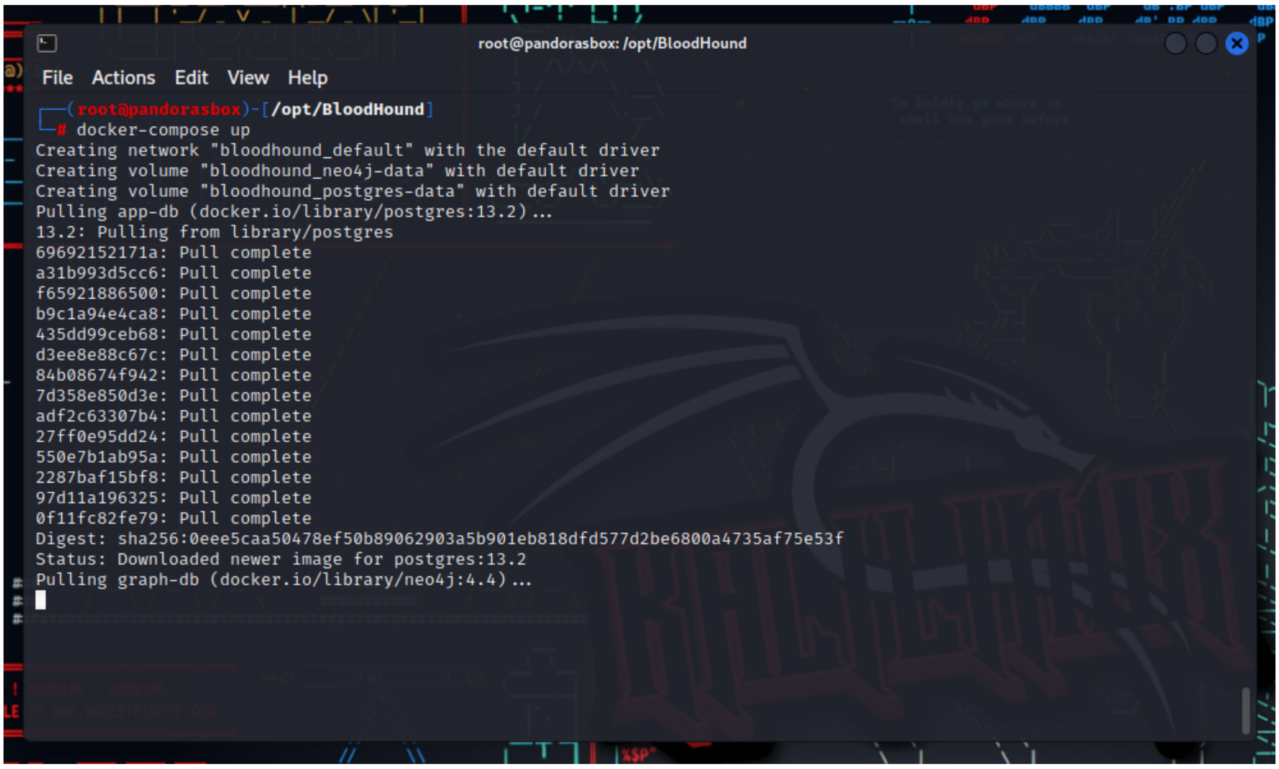
```
git clone https://github.com/SpecterOps/BloodHound.git /opt/BloodHound
cd /opt/BloodHound/
cp examples/docker-compose/* ./
```

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

As stated before, for a single user initial deployment, this will give us everything we need. We can start BloodHound with the following command:

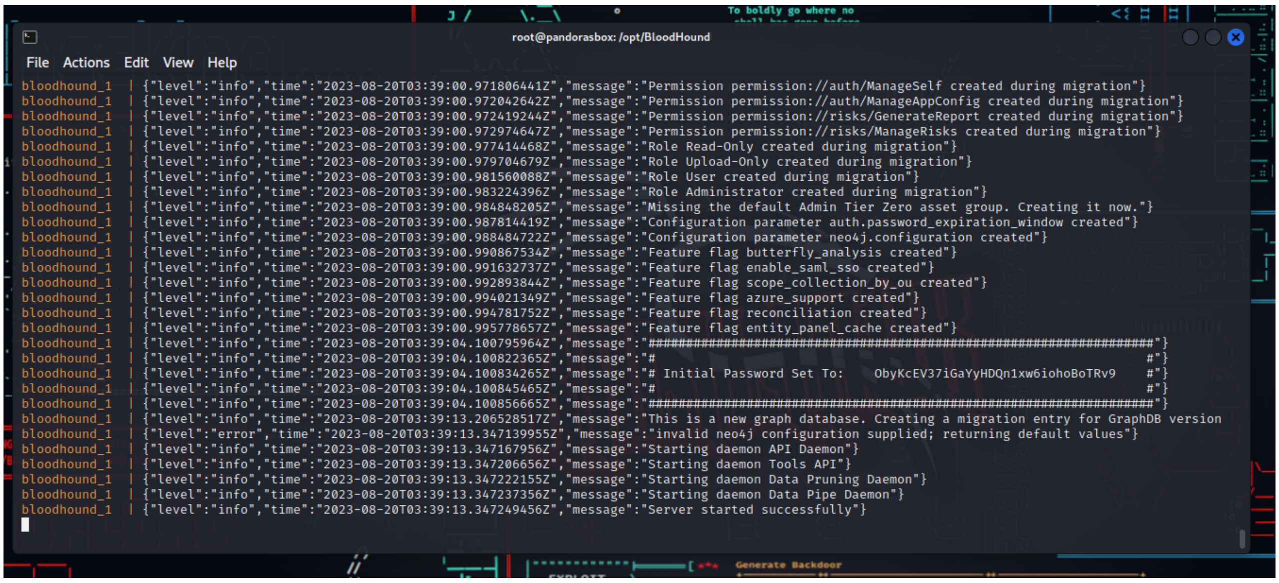
```
docker-compose up
```

This will then begin downloading the required containers from docker.io, please note this may take a few minutes.


root@pandorasbox: /opt/BloodHound
File Actions Edit View Help

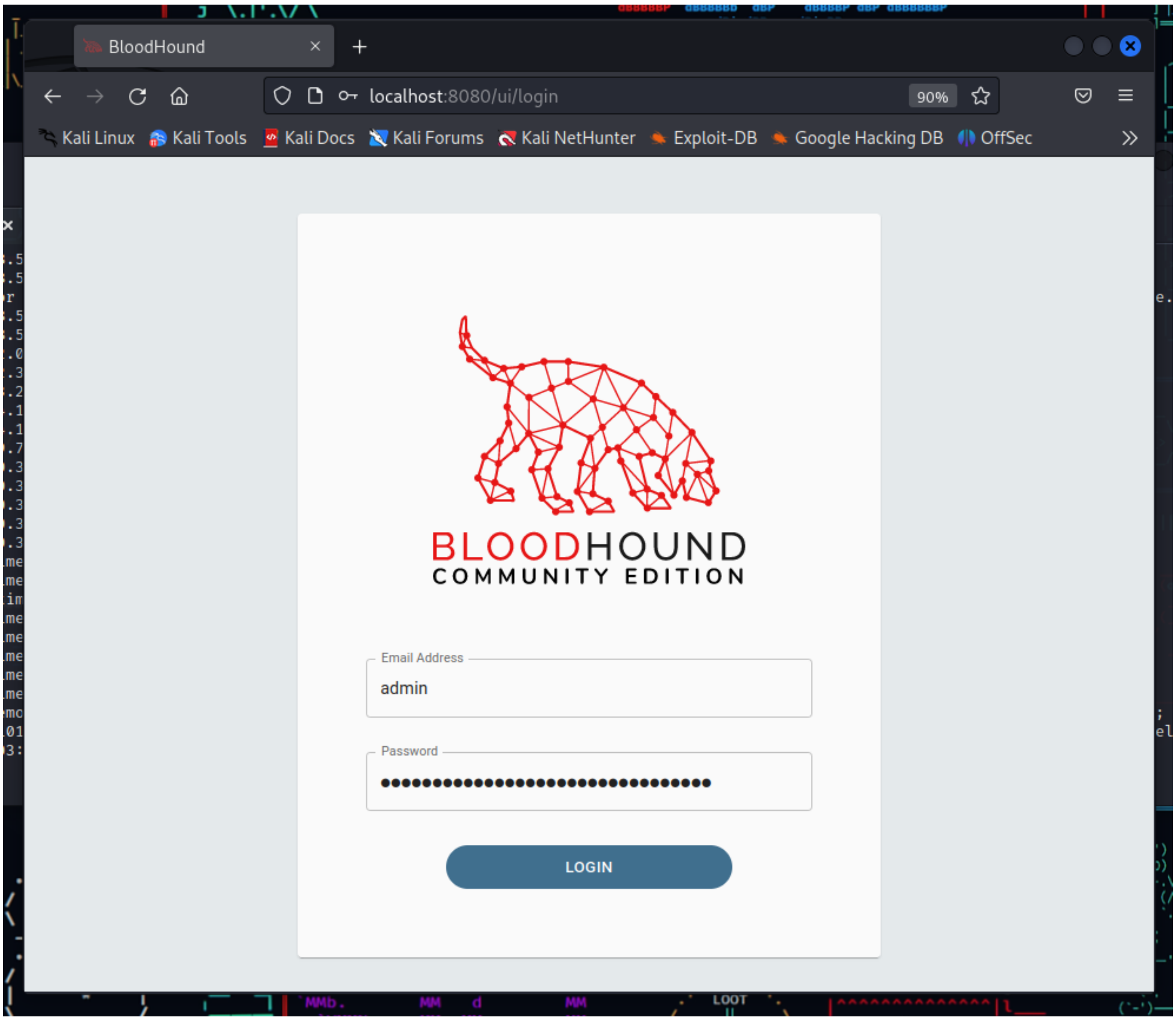
```
┌──(root@pandorasbox)-[/opt/BloodHound]
└─# docker-compose up
Creating network "bloodhound_default" with the default driver
Creating volume "bloodhound_neo4j-data" with default driver
Creating volume "bloodhound_postgres-data" with default driver
Pulling app-db (docker.io/library/postgres:13.2)...
13.2: Pulling from library/postgres
69692152171a: Pull complete
a31b993d5cc6: Pull complete
f65921886500: Pull complete
b9c1a94e4ca8: Pull complete
435dd99ceb68: Pull complete
d3ee8e88c67c: Pull complete
84b08674f942: Pull complete
7d358e850d3e: Pull complete
adf2c63307b4: Pull complete
27ff0e95dd24: Pull complete
550e7b1ab95a: Pull complete
2287baf15bf8: Pull complete
97d11a196325: Pull complete
0f11fc82fe79: Pull complete
Digest: sha256:0eee5caa50478ef50b89062903a5b901eb818dfd577d2be6800a4735af75e53f
Status: Downloaded newer image for postgres:13.2
Pulling graph-db (docker.io/library/neo4j:4.4)...
```

The Deployment will be complete when you see an initial password displayed on the screen. This will be the password for the Administrator user account.

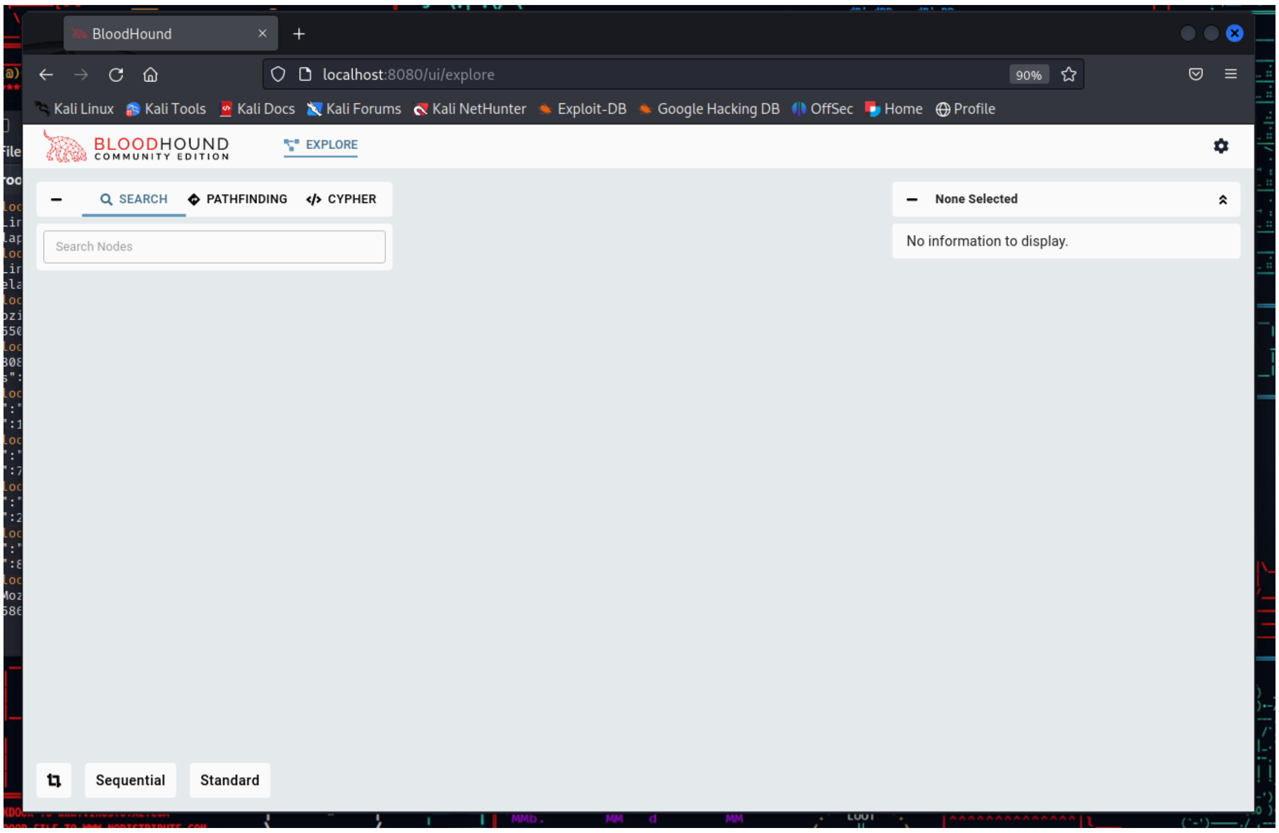


```
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.971806441Z","message":"Permission permission://auth/ManageSelf created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.972042642Z","message":"Permission permission://auth/ManageAppConfig created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.972419244Z","message":"Permission permission://risks/GenerateReport created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.972974647Z","message":"Permission permission://risks/ManageRisks created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.977414468Z","message":"Role Read-Only created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.979704679Z","message":"Role Upload-Only created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.981560088Z","message":"Role User created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.983224396Z","message":"Role Administrator created during migration"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.984848205Z","message":"Missing the default Admin Tier Zero asset group. Creating it now."}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.987814419Z","message":"Configuration parameter auth.password_expiration_window created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.988484722Z","message":"Configuration parameter neo4j.configuration created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.990867534Z","message":"Feature flag butterfly_analysis created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.991632737Z","message":"Feature flag enable_saml_sso created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.992893844Z","message":"Feature flag scope_collection_by_ou created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.994021349Z","message":"Feature flag azure_support created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.994781752Z","message":"Feature flag reconciliation created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:00.995778657Z","message":"Feature flag entity_panel_cache created"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:04.100795964Z","message":"###################################################################"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:04.100822365Z","message":"#                                                                 #"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:04.100834265Z","message":"# Initial Password Set To:    ObyKcEV37iGaYyHDQn1xw6iohoBoTRv9    #"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:04.100845465Z","message":"#                                                                 #"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:04.100856665Z","message":"###################################################################"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.206528517Z","message":"This is a new graph database. Creating a migration entry for GraphDB version
bloodhound_1  | {"level":"error","time":"2023-08-20T03:39:13.347139955Z","message":"invalid neo4j configuration supplied; returning default values"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.347167956Z","message":"Starting daemon API Daemon"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.347206656Z","message":"Starting daemon Tools API"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.347222155Z","message":"Starting daemon Data Pruning Daemon"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.347237356Z","message":"Starting daemon Data Pipe Daemon"}
bloodhound_1  | {"level":"info","time":"2023-08-20T03:39:13.347249456Z","message":"Server started successfully"}
```

You will now be able to login with the initial set password.



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



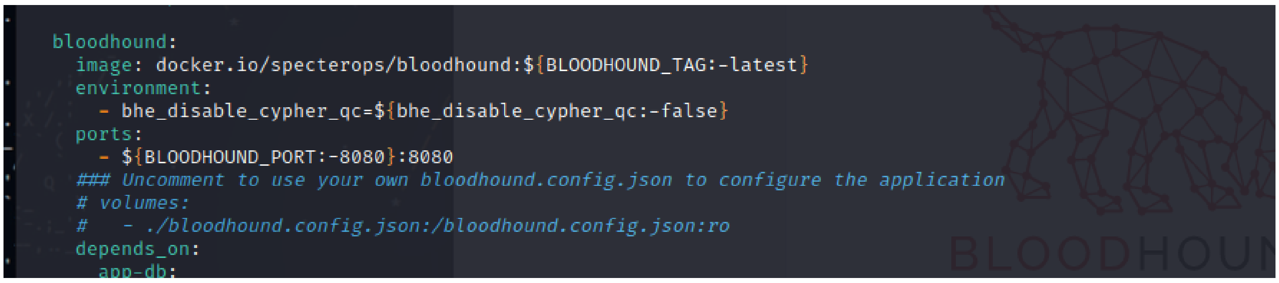
Modifying the BloodHound and Docker Config Files

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.

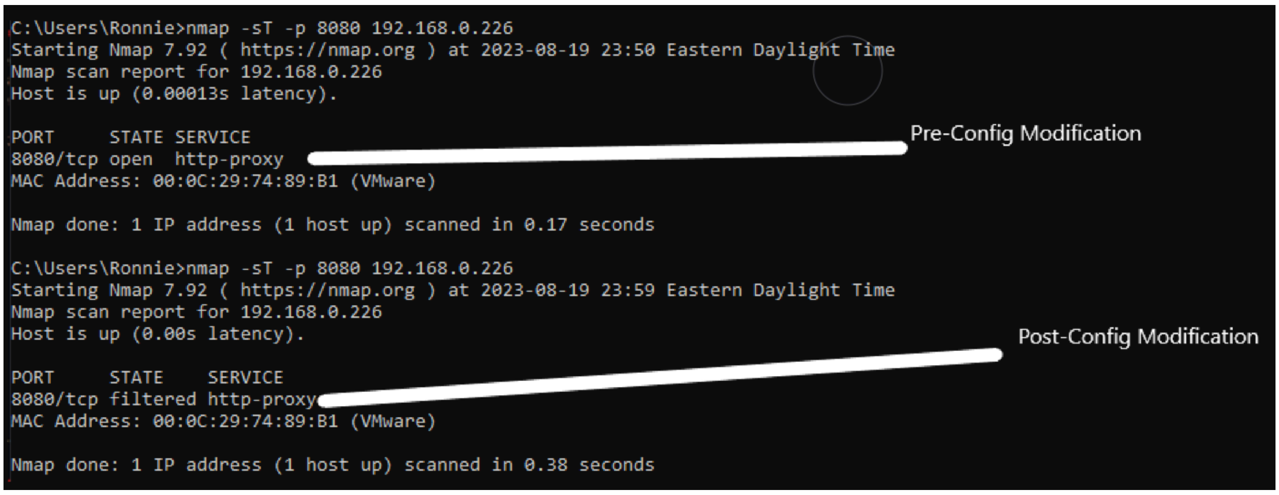
```
{
  "version": 1,
  "bind_addr": "127.0.0.1:8080",
  ...
    "last_name": "Admin",
    "email_address": "spam@example.com"
  }
}
```

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 `docker-compose up`, and the container will restart.



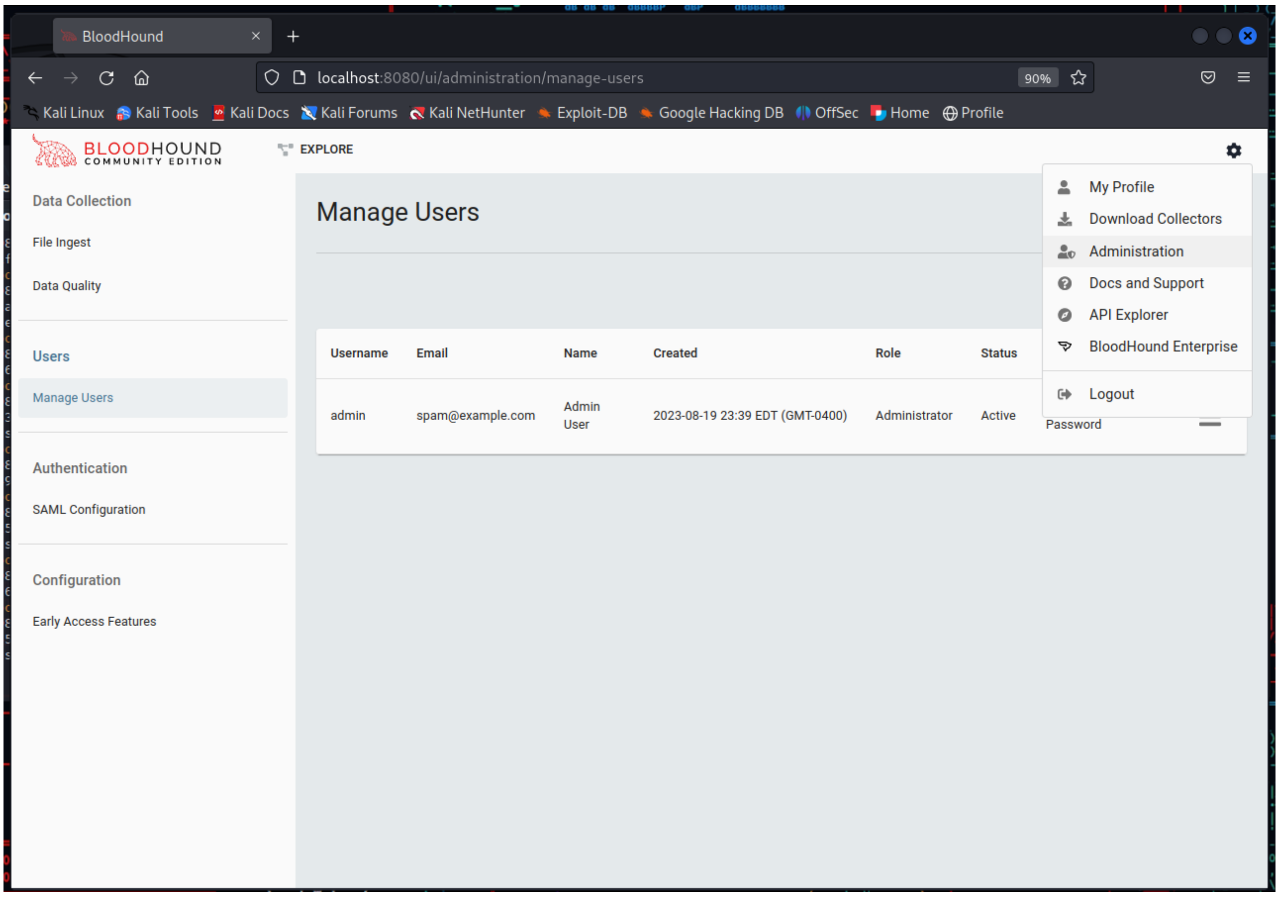
```
C:\Users\Ronnie>nmap -sT -p 8080 192.168.0.226
Starting Nmap 7.92 ( https://nmap.org ) at 2023-08-19 23:50 Eastern Daylight Time
Nmap scan report for 192.168.0.226
Host is up (0.00013s latency).

PORT     STATE SERVICE
8080/tcp open  http-proxy
MAC Address: 00:0C:29:74:89:B1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds

C:\Users\Ronnie>nmap -sT -p 8080 192.168.0.226
Starting Nmap 7.92 ( https://nmap.org ) at 2023-08-19 23:59 Eastern Daylight Time
Nmap scan report for 192.168.0.226
Host is up (0.00s latency).

PORT     STATE    SERVICE
8080/tcp filtered http-proxy
MAC Address: 00:0C:29:74:89:B1 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.38 seconds
```

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.

```
{
  "version": 1,
  "bind_addr": "0.0.0.0:8080",
...
    "last_name": "Admin",
    "email_address": "spam@example.com"
  }
}
```

Creating User Accounts

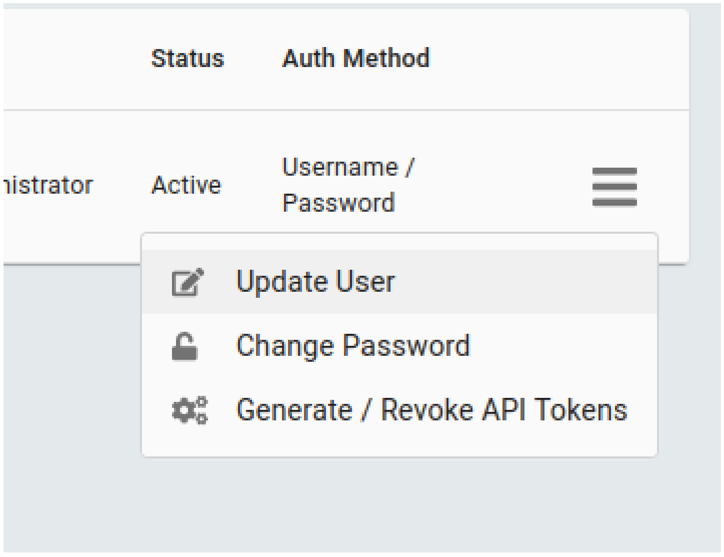
As always, in a team setting it's bad practice to use the default Administrator user as your personal user account, so in this next section, we'll show you how to create a new user. We can do this by navigating to the settings cog, selecting "Administration", then "Manage Users".



Manage Users

Username	Email	Name	Created	Role	Status	
admin	spam@example.com	Admin User	2023-08-19 23:39 EDT (GMT-0400)	Administrator	Active	Password

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”.

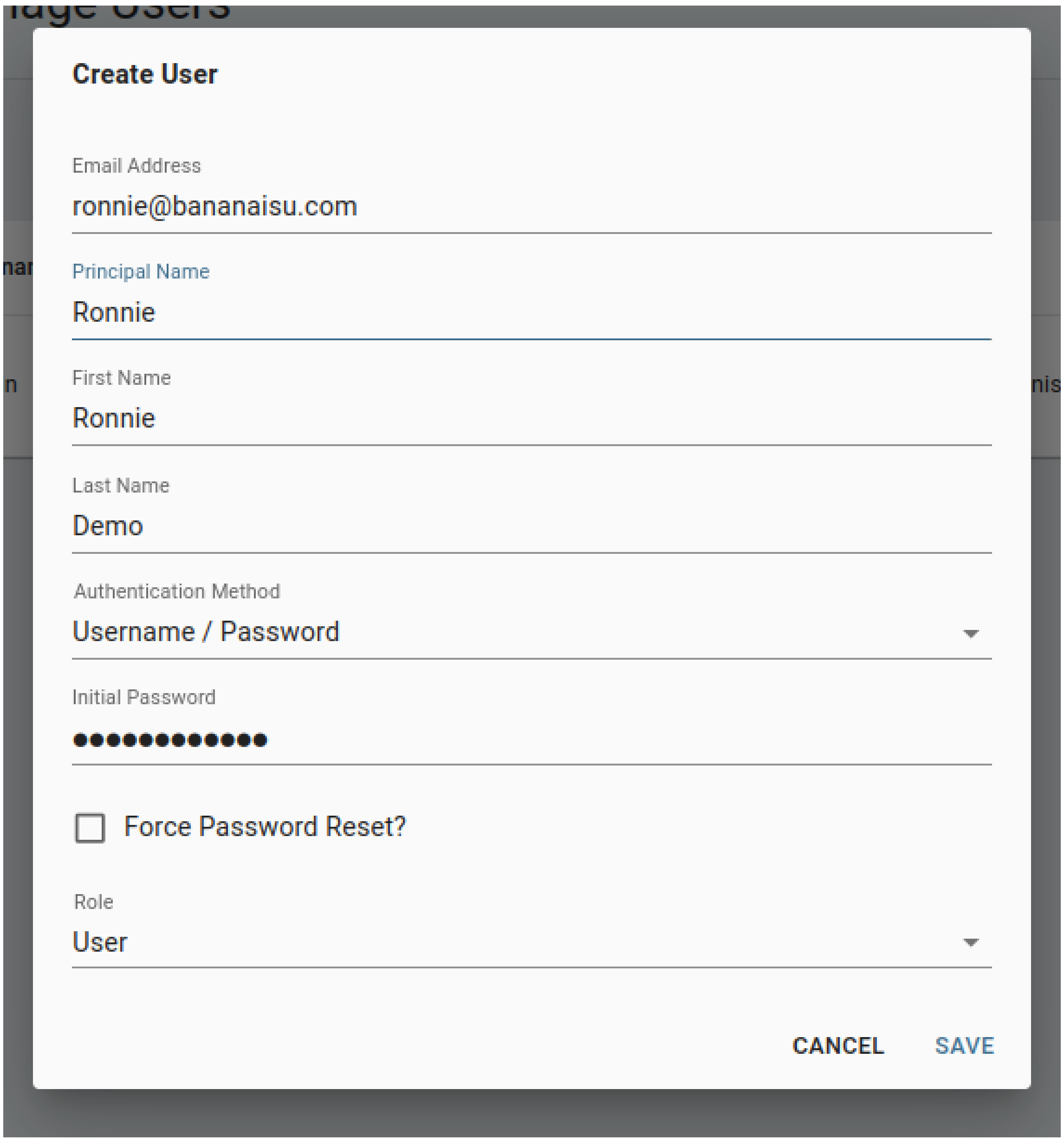


	Status	Auth Method
istrator	Active	Username / Password

Additionally, this is also modifiable in the BloodHound config file:

```
...
  "collectors_base_path":
"/etc/bloodhound/collectors",
  "default_admin": {
    "principal_name": "admin",
    "first_name": "Bloodhound",
    "last_name": "Admin",
    "email_address":
"spam@example.com"
  }
}
```

Back to the web application,
selecting “Create User” will open a new dialogue box:


Create User

Email Address
ronnie@bananaisu.com

Principal Name
Ronnie

First Name
Ronnie

Last Name
Demo

Authentication Method
Username / Password

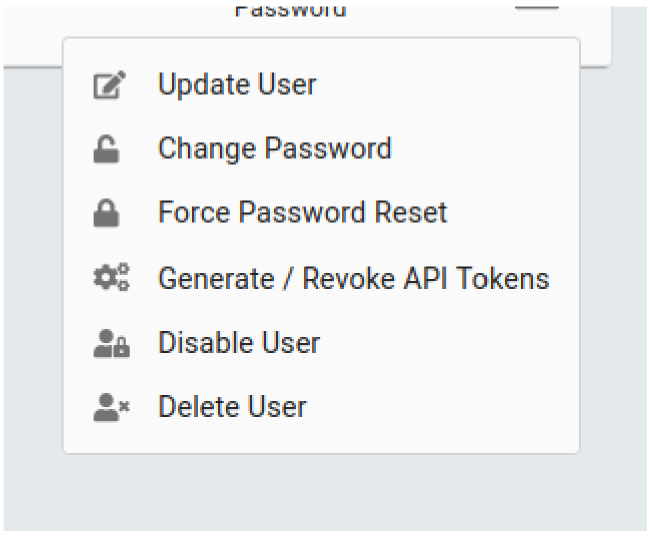
Initial Password
●●●●●●●●●●●●

☐ Force Password Reset?

Role
User

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.

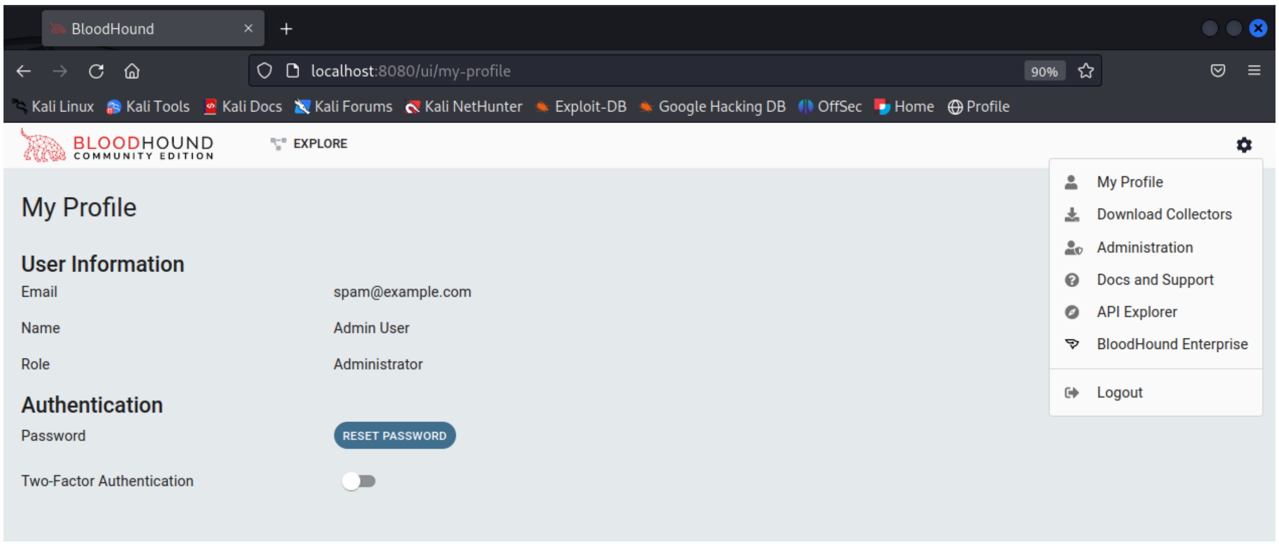
For awareness, it's possible to disable, force password changes, change passwords, generate API tokens, and delete user accounts as well.



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".



My Profile

User Information

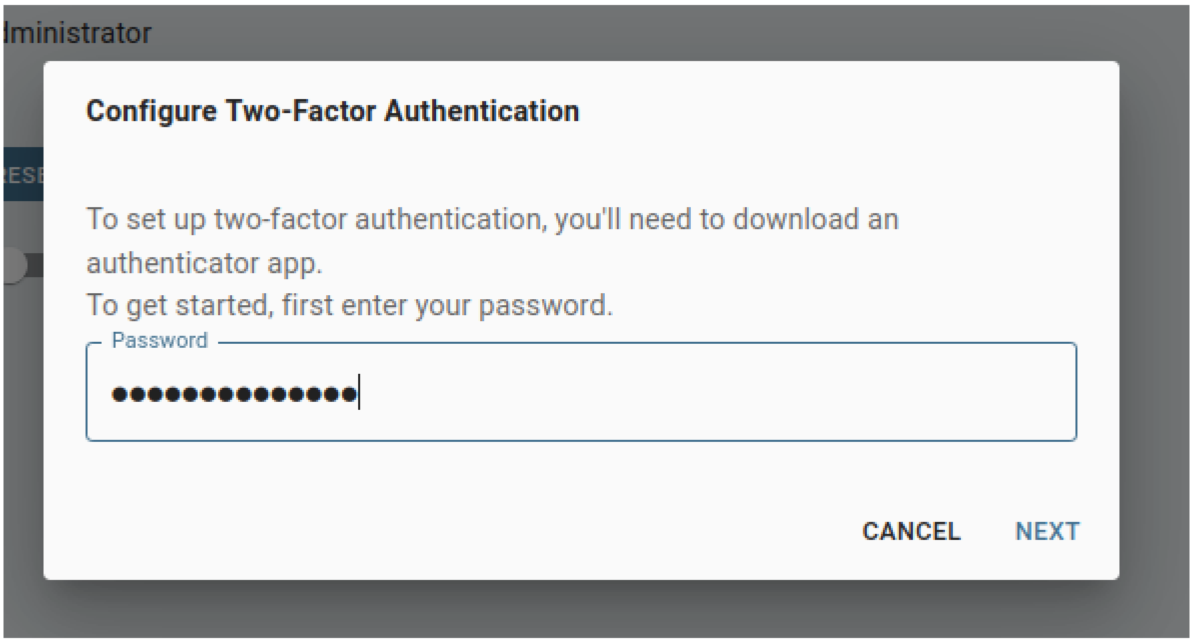
Email	spam@example.com
Name	Admin User
Role	Administrator

Authentication

Password

Two-Factor Authentication

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

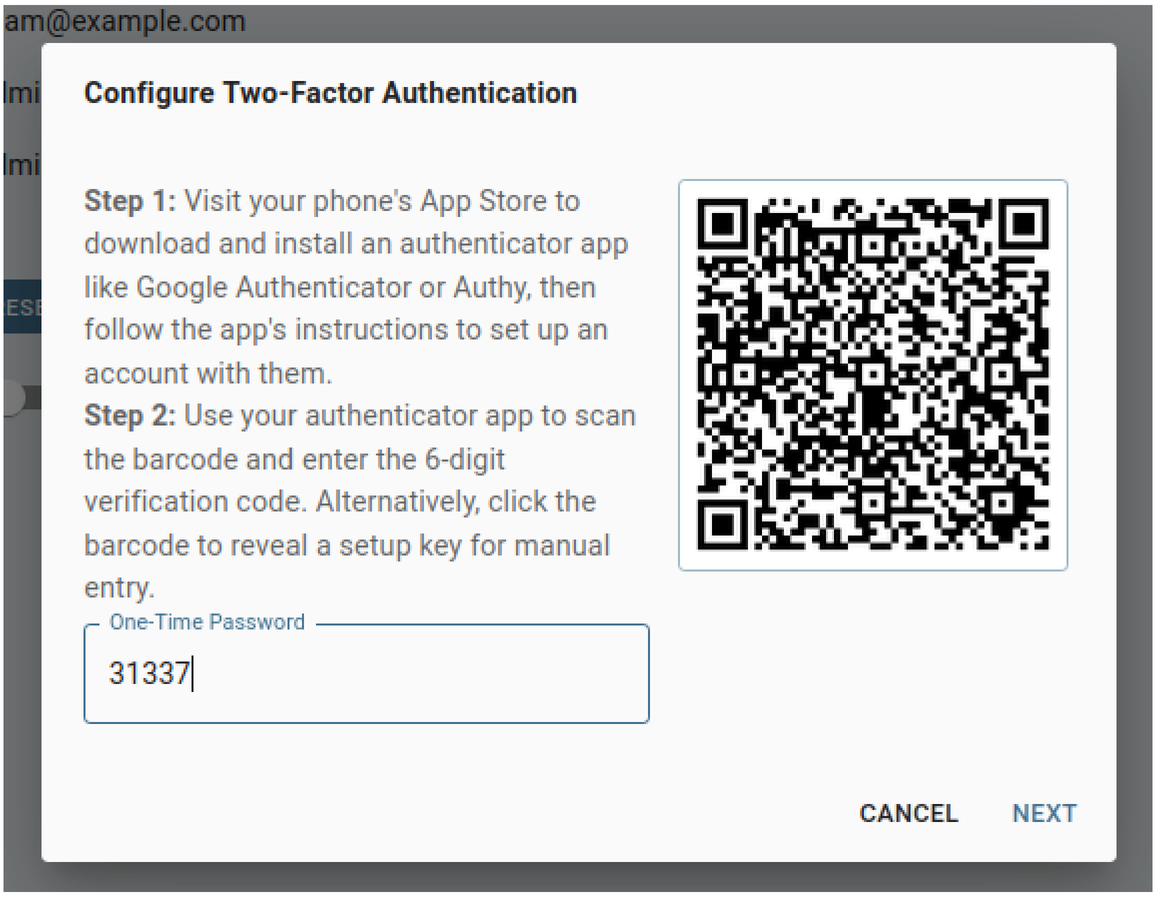

lministrator

Configure Two-Factor Authentication

To set up two-factor authentication, you'll need to download an authenticator app.
To get started, first enter your password.

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.

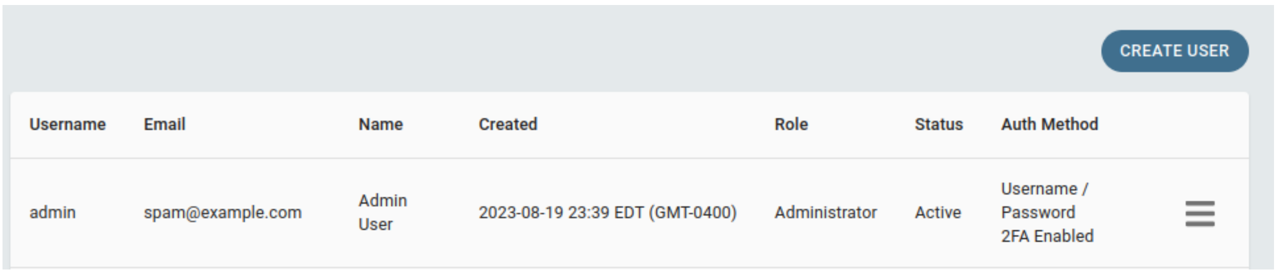

am@example.com

Configure Two-Factor Authentication

Step 1: Visit your phone's App Store to download and install an authenticator app like Google Authenticator or Authy, then follow the app's instructions to set up an account with them.
Step 2: Use your authenticator app to scan the barcode and enter the 6-digit verification code. Alternatively, click the barcode to reveal a setup key for manual entry.

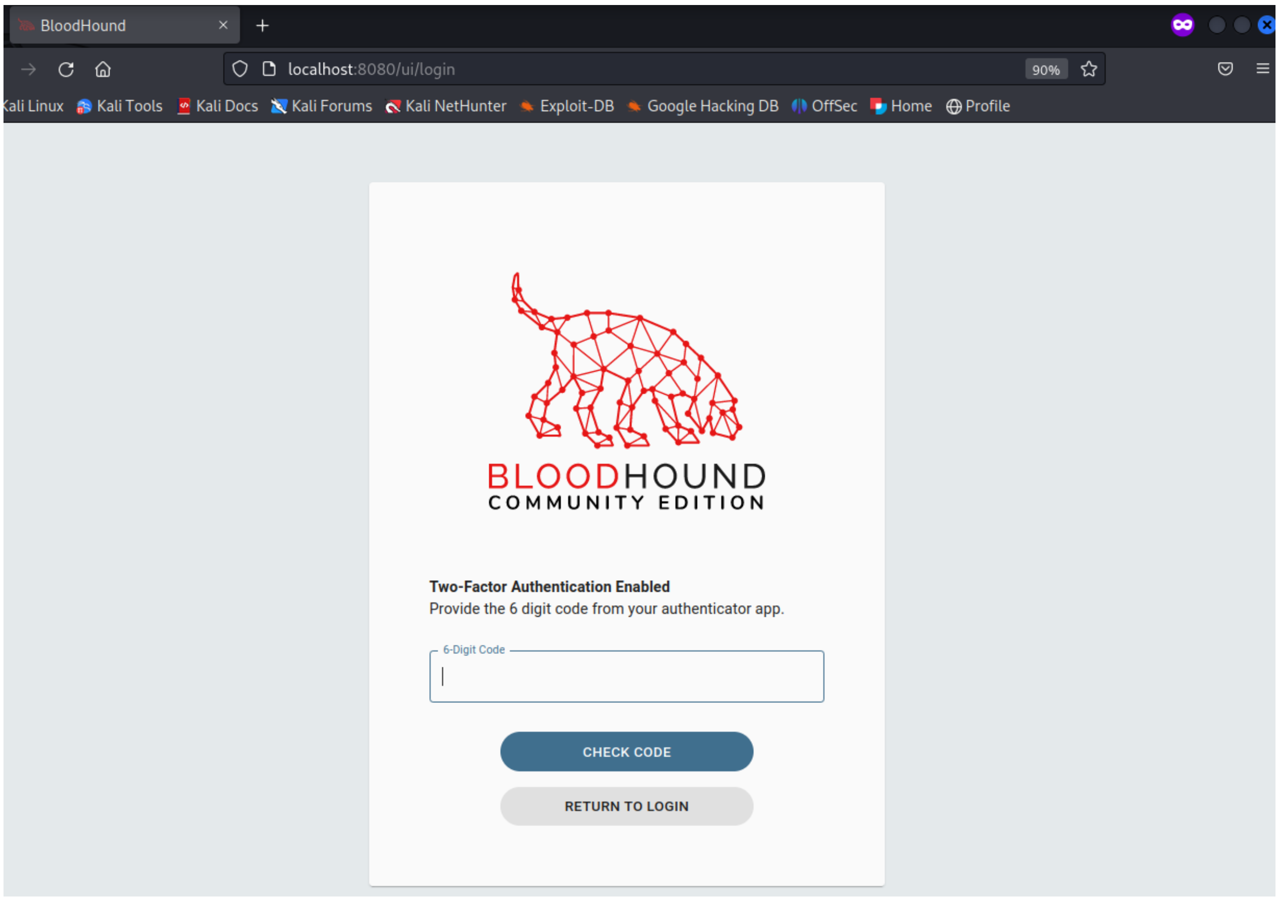
One-Time Password: 31337

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.



Username	Email	Name	Created	Role	Status	Auth Method
admin	spam@example.com	Admin User	2023-08-19 23:39 EDT (GMT-0400)	Administrator	Active	Username / Password 2FA Enabled

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



Two-Factor Authentication Enabled
Provide the 6 digit code from your authenticator app.

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.

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

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

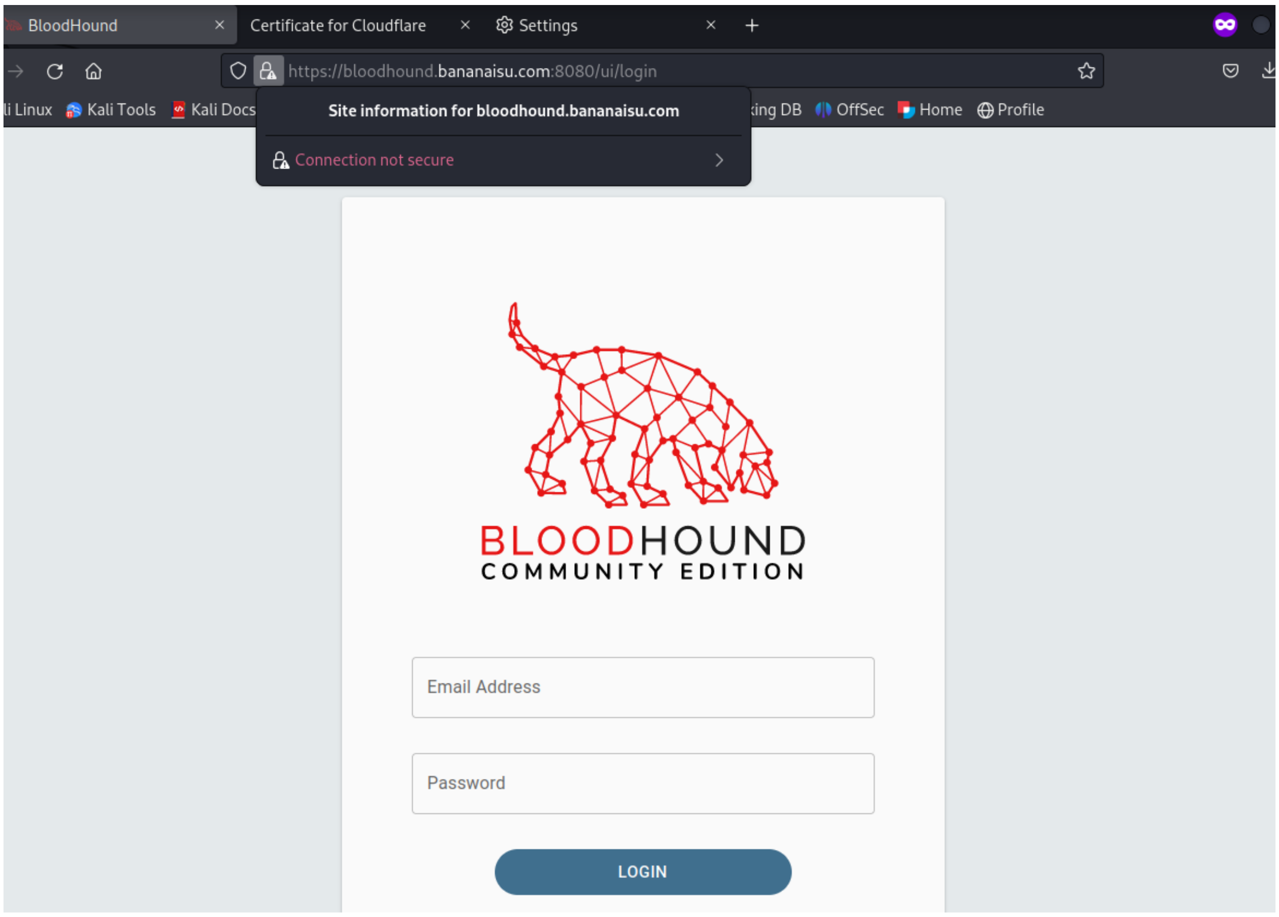
```
bloodhound_1  | {"level":"error","time":"2023-08-
20T04:50:35.429379169Z","message":"HTTP server listen error: open
/opt/bloodhound/cert.cer: no such file or directory"}
bloodhound_1  | {"level":"error","time":"2023-08-
20T04:50:35.429503069Z","message":"HTTP server listen error: open
/opt/bloodhound/cert.cer: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

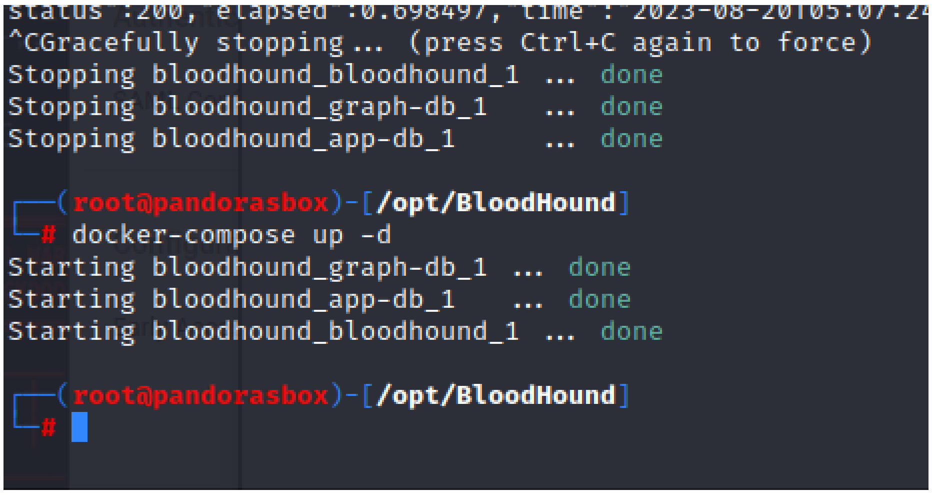
Backgrounding Docker

This is our last section - Backgrounding our docker container. We've made a lot of good configuration changes. Don't be like me, go away for lunch, shut the laptop and one of your coworkers tries to access the BHCE instance and it's not working. Why's that? The SSH connection to your BHCE server was terminated!

To prevent this, we're going to detach the container by stopping the container and restarting it with the following command:

```
docker-compose up -d
```

This will start BHCE as a backgrounded process:



```
status":200,"elapsed":0.698497,"time":"2023-08-20T05:07:24
^CGracefully stopping... (press Ctrl+C again to force)
Stopping bloodhound_bloodhound_1 ... done
Stopping bloodhound_graph-db_1   ... done
Stopping bloodhound_app-db_1     ... done

┌──(root@pandorasbox)-[/opt/BloodHound]
└─# docker-compose up -d
Starting bloodhound_graph-db_1 ... done
Starting bloodhound_app-db_1   ... done
Starting bloodhound_bloodhound_1 ... done

┌──(root@pandorasbox)-[/opt/BloodHound]
└─#
```

If you make a change and need to restart one of the containers, or shut them down for some reason, you can run the following command to shutdown the container:

```
docker-compose down
```

I recommend creating some aliases to do this just to expedite the process, especially if you're not a Docker power-user. This is my first real experience with using Docker and it's been a bit of a struggle, so I thought I'd share some of the things I learned just to help make your all's life a little bit easier.

Anyways, I hope this helps :D

~ Ronnie

Comments