SecureRS

Email: asingh@cs.up.ac.za
Website: https://github.com/AvinashSingh786/SecureRS

User Manual

DigiForS
Digital Forensic Science

About

This tool was designed for research in the field of Digital Forensics.

This prototype solution was created with Digital Forensic Readiness processes for secure storage and retrieval or potential digital evidence. This solution is generic and can be used for any application that requires secure storage. There is also an API built in that allows integration with any system or tool. From the admin panel you can create and manage API keys and routes.

Installation

This tool can be run from a docker container that can be built using the dockerfile. Alternatively, you can clone this repository and install the python requirements. This tool only works for Python3 and was tested with Python3.7. It is recommended you run this in a virtual environment to further ensure compatibility and added security.

```
$ git clone git@github.com:AvinashSingh786/SecureRS.git
$ cd SecureRS
$ python3 -m pip install --user virtualenv
$ apt-get install python3-venv python3-magic # for Linux
$ python3 -m venv venv
$ source env/bin/activate # for Linux
$ pip3 install python-magic-bin # for Windows
$ .\env\Scripts\activate # for Windows
(venv)$ pip3 install -r requirements.txt
```

Usage

Run the following commands to configure and run the engine.

```
(venv)$ python3 manage.py makemigrations pde # This sets up the storage engine
(venv)$ python3 manage.py migrate # This creates the databases and interfaces
(venv)$ python3 manage.py createsuperuser # Create a super user that you will use
```

Features

- OTP Login + Download (TOTP, YubiKey)
- REST API for Ingestion
- Two Factor Auth
- Secure Cookies
- Integrity Verification
- Encrypted Storage
- Security Headers
- Email Config
- Session Security
- Customizable

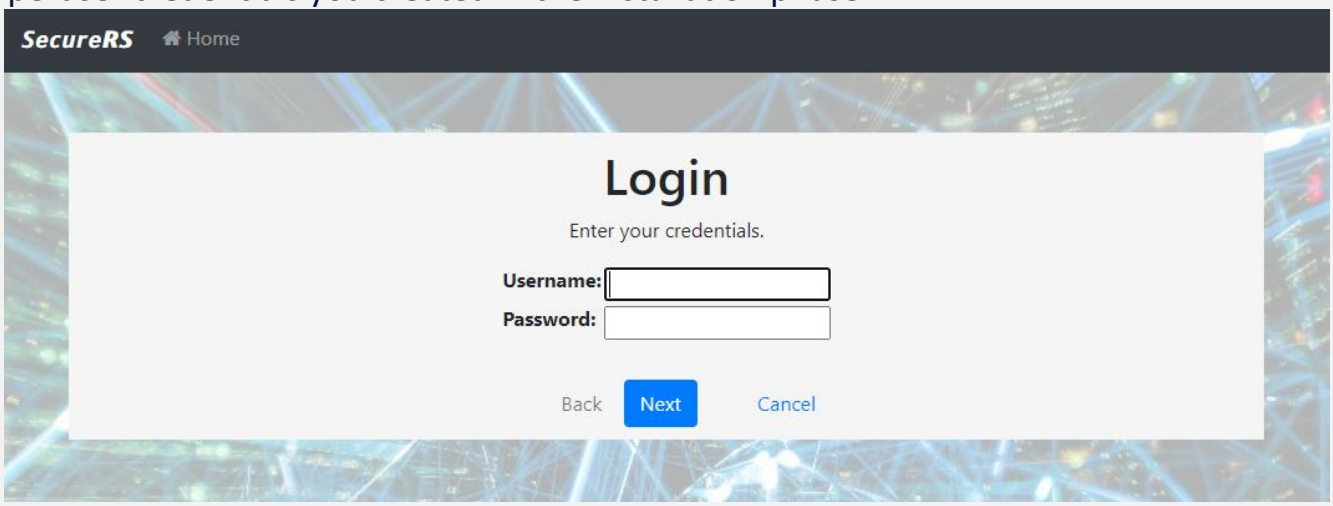
Important!

If you plan on using this tool in production, please change the following in the settings.py file:

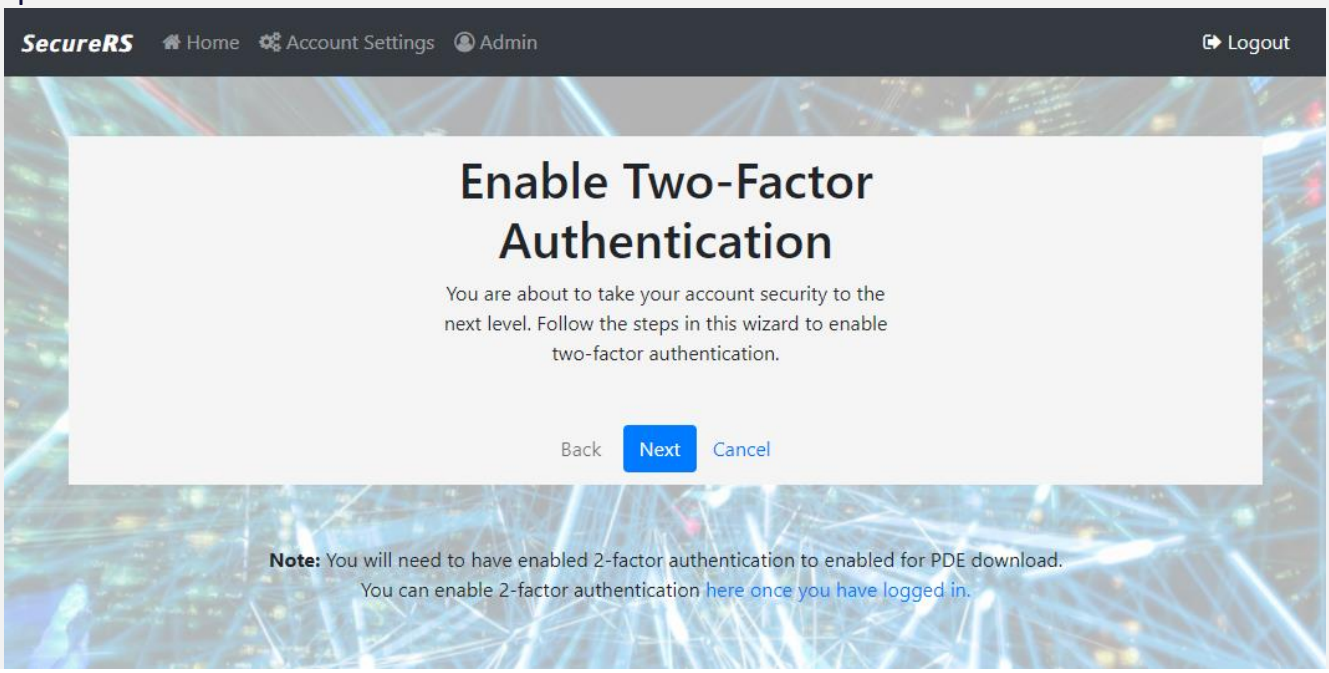
- SECRET_KEY
- DEBUG
- ALLOWED_HOSTS
- COMPANY_NAME
- DEFF_PASSWORD
- DEFF_SALT
- SESSION_SECURITY_EXPIRE_AFTER
- SESSION_SECURITY_WARN_AFTER
- EMAIL_USE_TLS
- EMAIL_HOST
- EMAIL_PORT
- EMAIL_HOST_USER
- EMAIL_HOST_PASSWORD

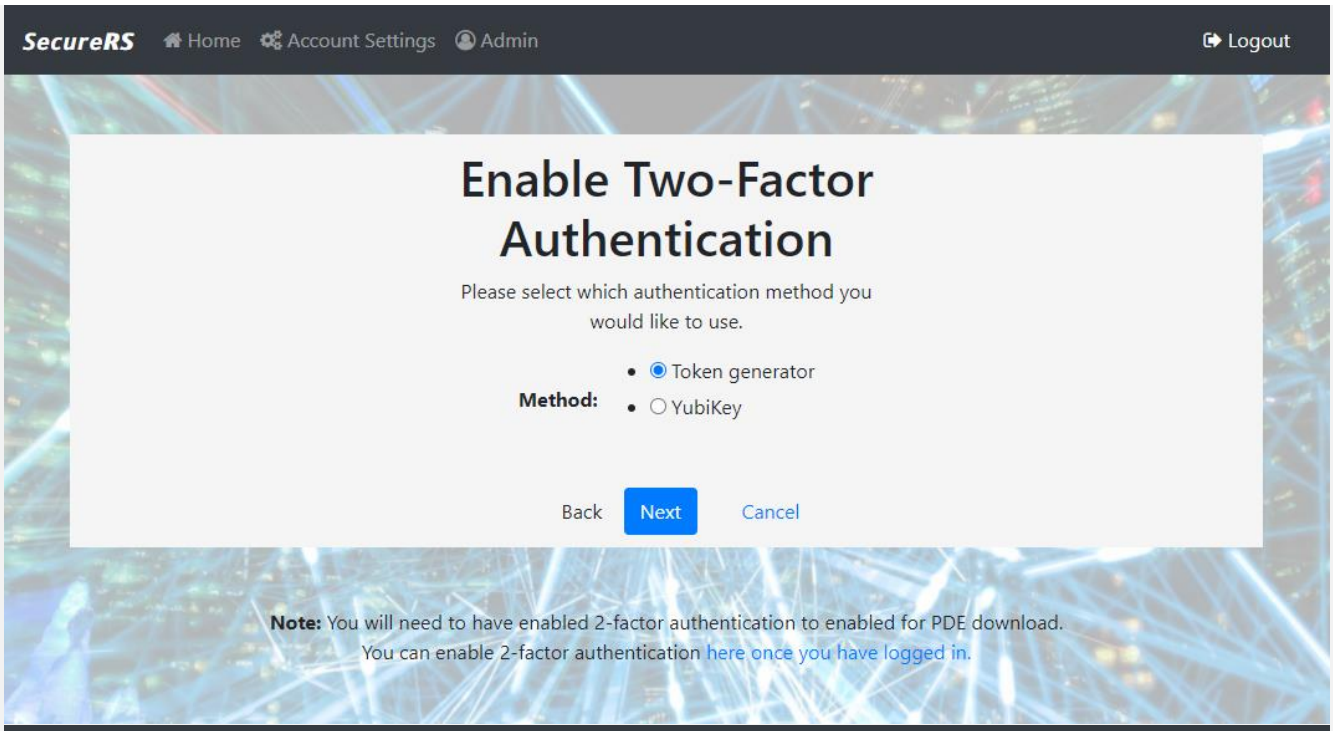
Login

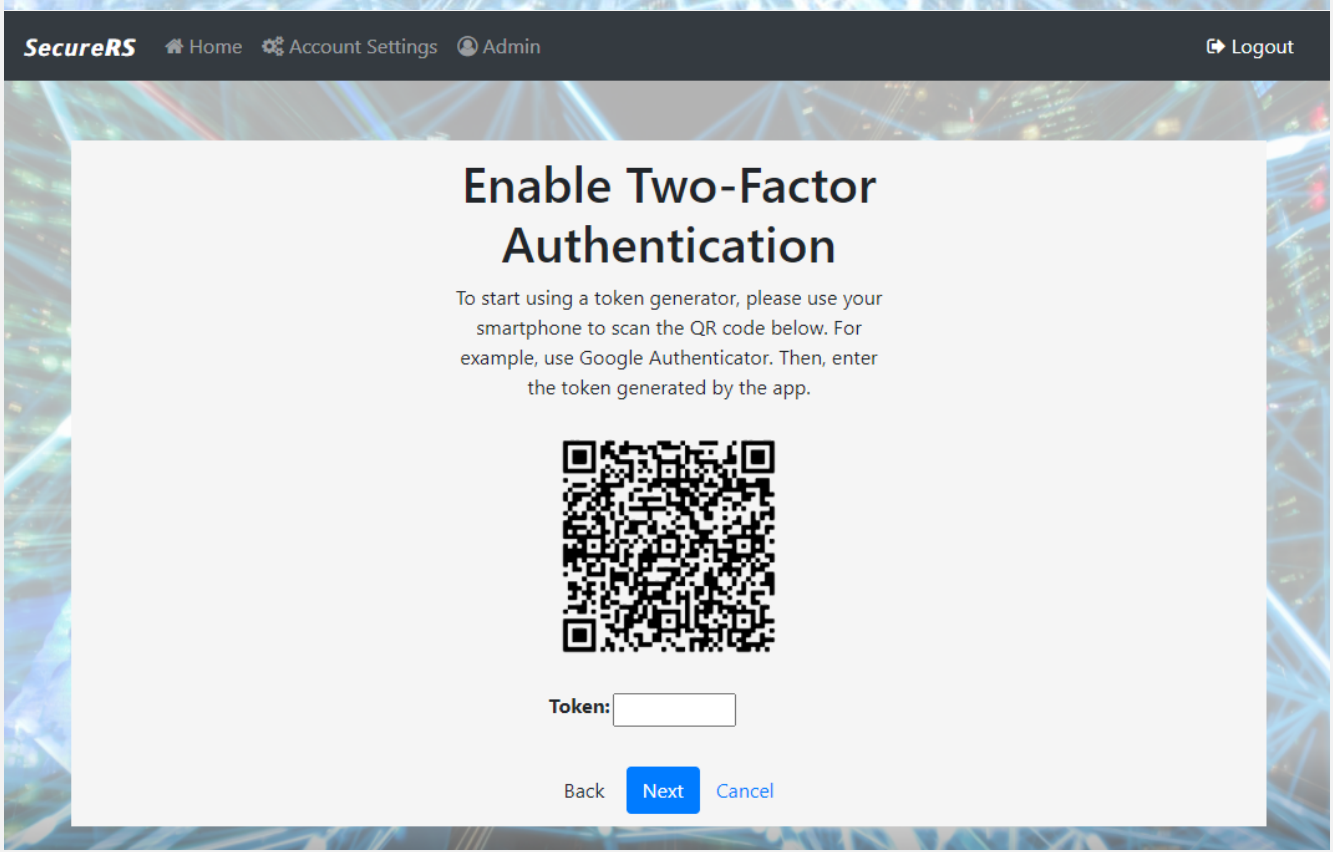
Once you have installed and setup the system you can navigate to https://localhost:8000 and will be presented with the screenshot below. If you are using a self-signed SSL certificate you will see browser errors which you can ignore for testing purposes. Login with the superuser credentials you created in the installation phase.



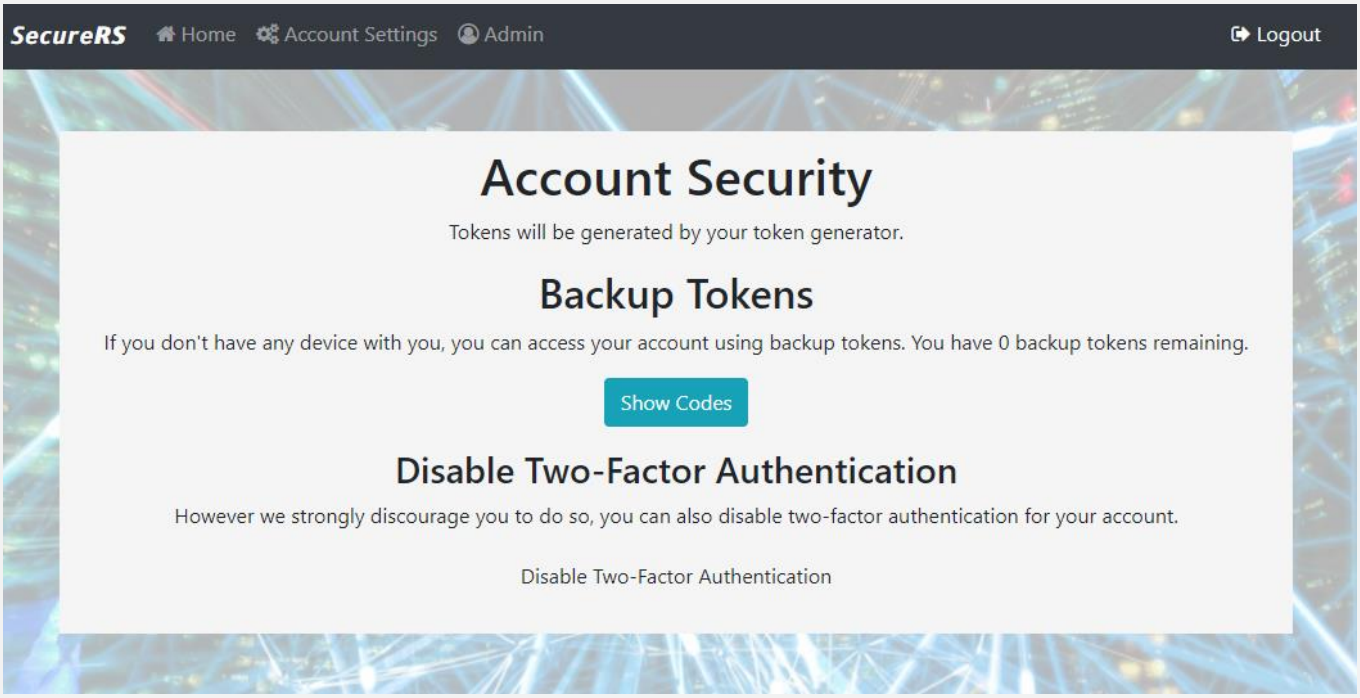
After you have logged in you will need to setup two-factor authentication by following the steps below.




SecureRS
Home
Account Settings
Admin
Logout
Enable Two-Factor Authentication
Please select which authentication method you would like to use.
Method:
Token generator
YubiKey
Back
Next
Cancel
Note: You will need to have enabled 2-factor authentication to enabled for PDE download.
You can enable 2-factor authentication here once you have logged in.

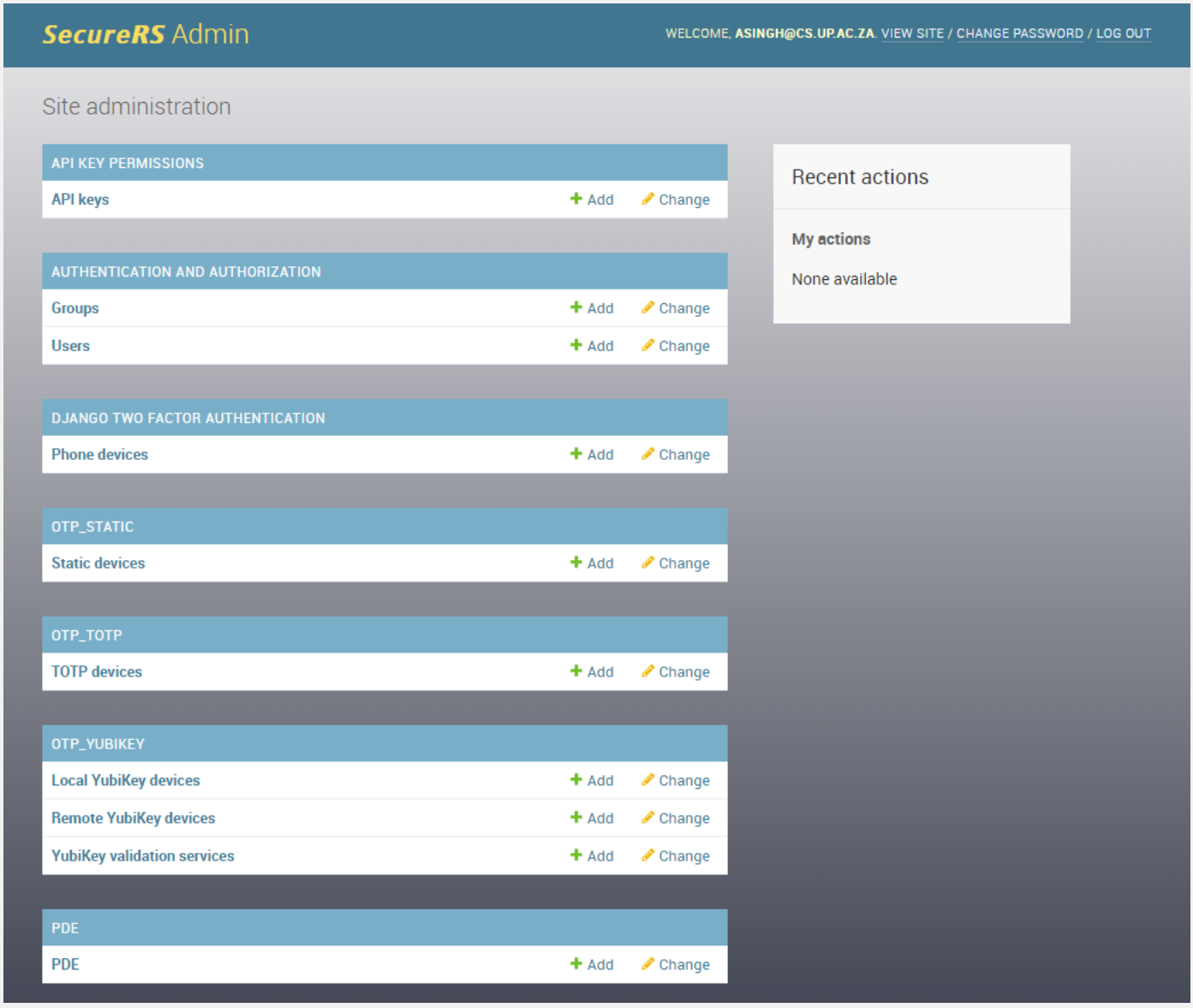

SecureRS
Home
Account Settings
Admin
Logout
Enable Two-Factor Authentication
To start using a token generator, please use your smartphone to scan the QR code below. For example, use Google Authenticator. Then, enter the token generated by the app.
Token:
Back
Next
Cancel

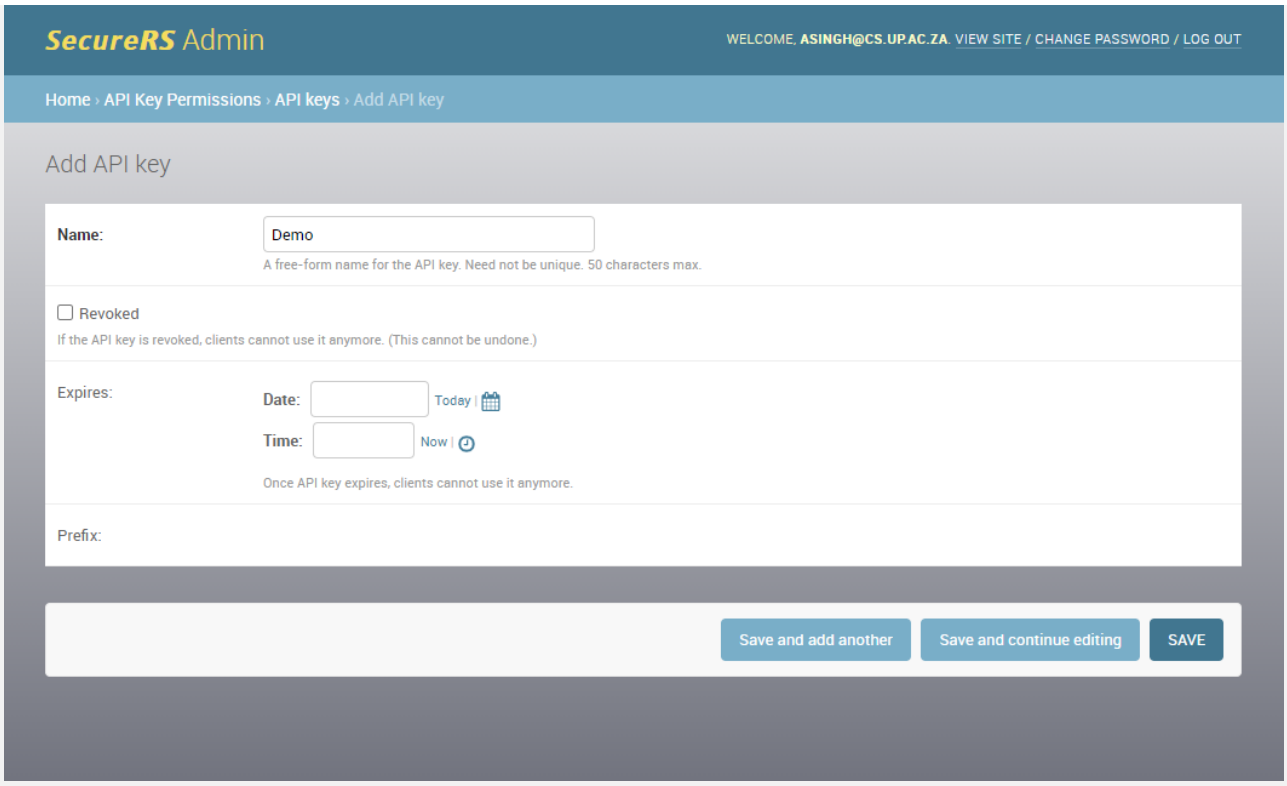
In the event you lose the 2FA app (Google Authenticator) or it resets, you should use the backup tokens. After each use of a backup token an email will be sent in order to protect your account, once a backup token has been used, it cannot be used again. You will need to enable 2FA in order to download any PDE files.



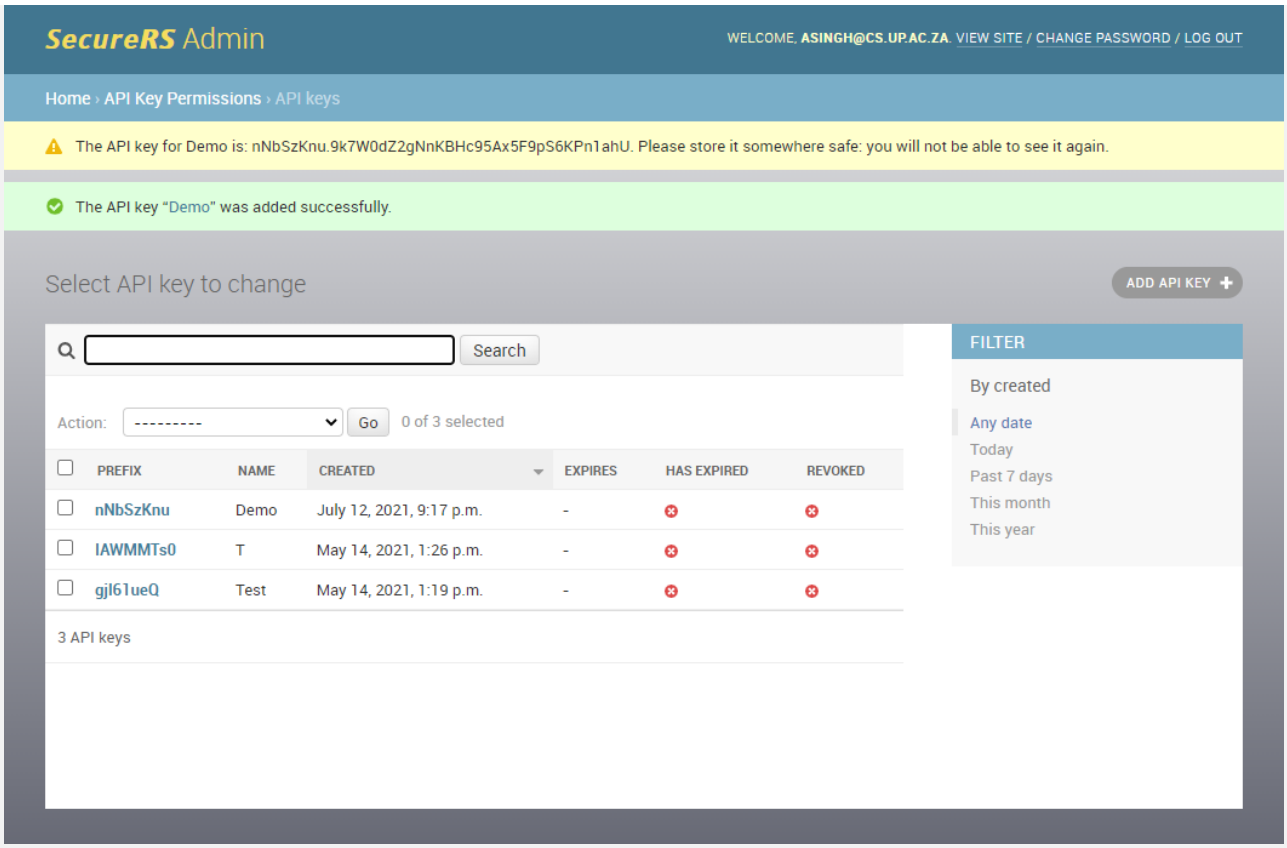
Admin

The Django Admin allows you to do a lot of powerful operations, one of which is management. Only superusers are able to view this, in order to add API keys. The next steps detail how an API key can be generated and added to the system.



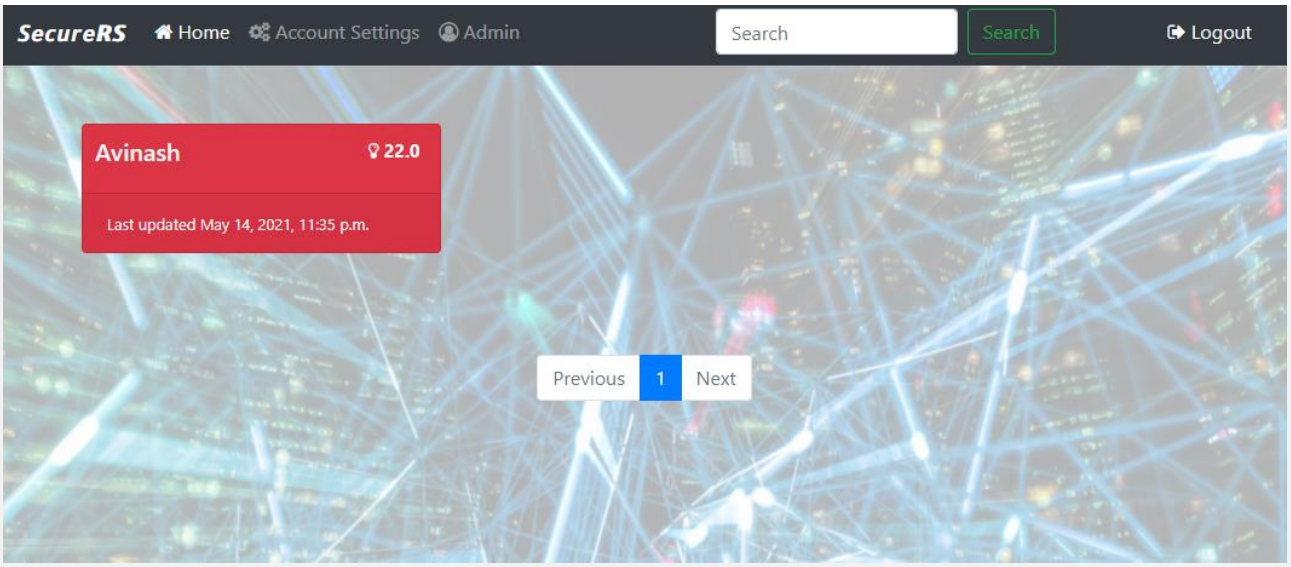


Once you have given the API key a name, you will be presented with the API Key, this will only be visible once and if the page is reloaded it will be forever lost, so ensure that you note it down. The platform also allows an API key to be revoked in the event the API key is no longer needed, but a record needs to be kept. This API key is then used for ingestion and any tool or application can used to submit data to be securely stored. This system was designed for Potential Digital Evidence (PDE) which are relatively small amounts of data, and is not meant for large disk dumps without performance penalty.

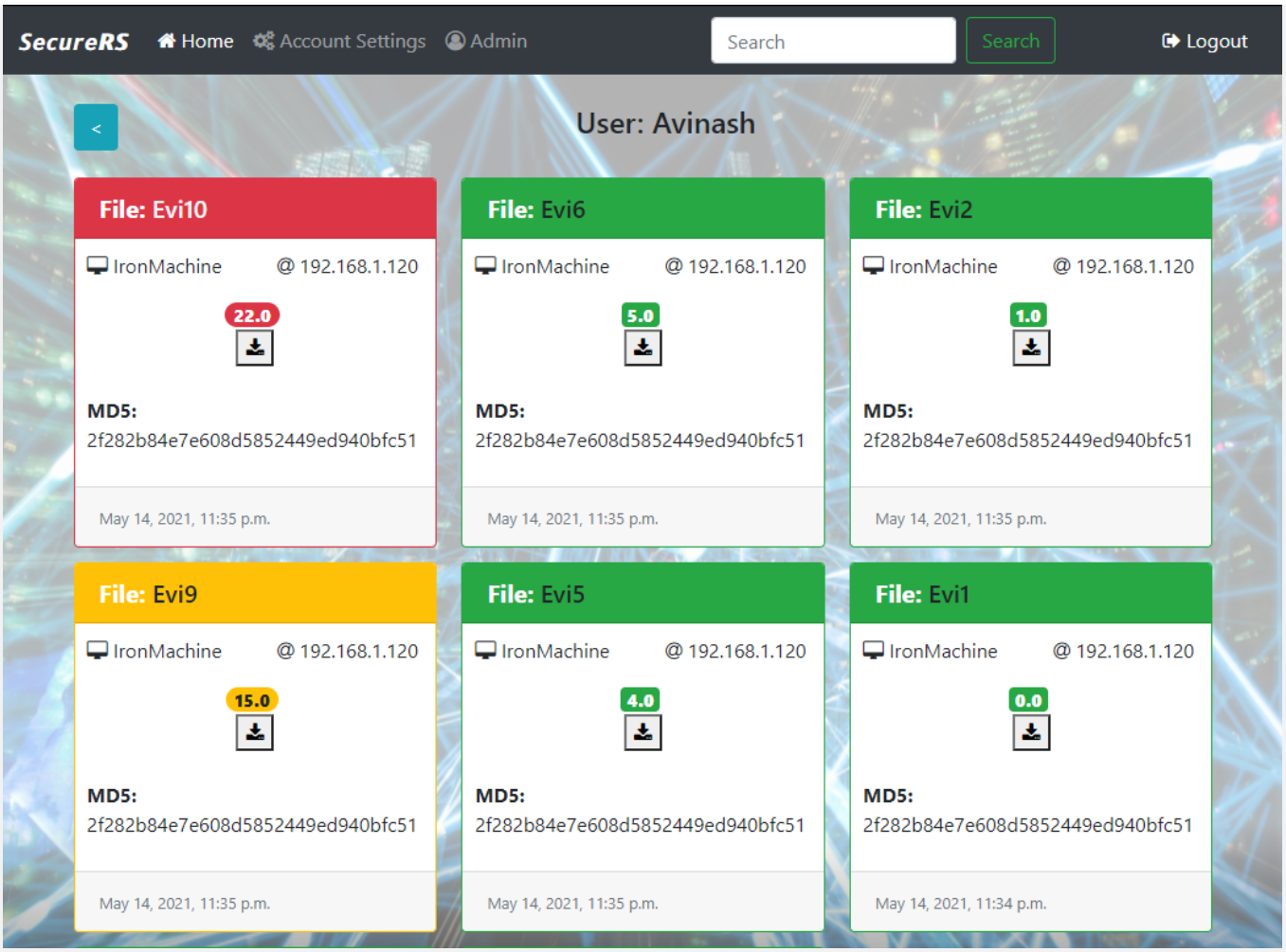


View/UI

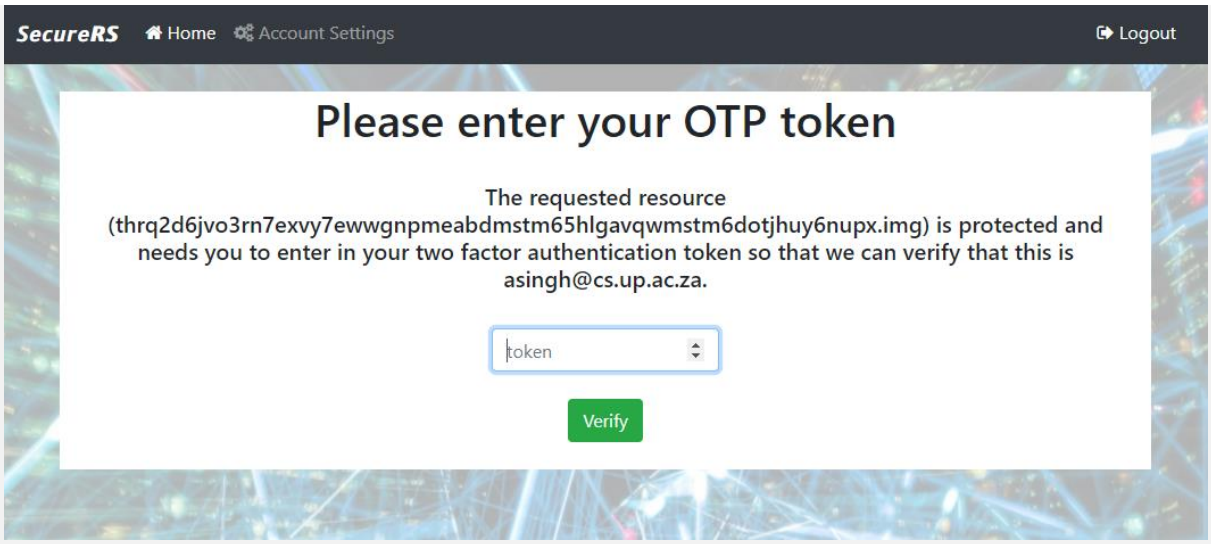
After the configuration and account setup has been completed any data submitted to the system can be seen by the superuser or investigator (through a separate standard account created in the Django Admin with limited permissions) and is separated by user for ease of access by an investigator. At a glance the latest PDE file is used to show if there are any latest issues in red which is configurable by the thresholds defined in the settings.py file.



Once the investigator has selected the correct user, all the PDE files stored for that specific user is then displayed with the rank, timestamp, md5 sum, machine name, ip address, date as well as the option to download the PDE.



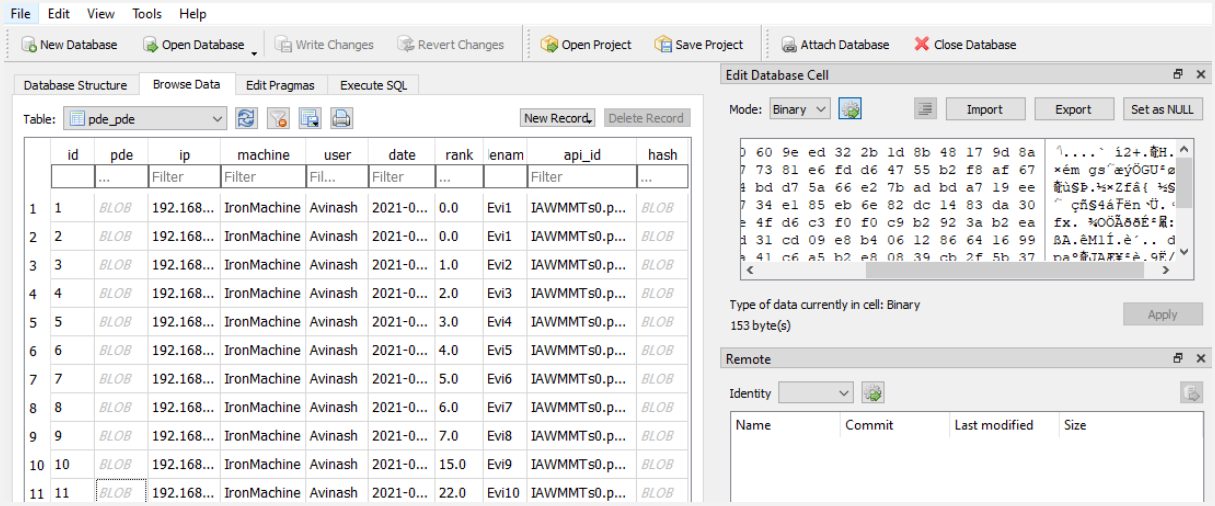
After the investigator has selected the PDE to download a 2FA screen will appear to validate the user as well as the privileges. Once the token is validated successfully an email will be sent with the MD5 hash, and the file will be decrypted and be downloaded to the local computer for further investigation. The last screenshot showcases the view of the database in which the Hash as well as the PDE file is stored as encrypted data in Blob format. This makes it impossible to alter the data without it being detected through the various forensic processes involved in this system.




```
Hi asingh@cs.up.ac.za,

You've verified yourself and just downloaded a PDE file with the following details:

File: thrq2d6jvo3rn7exvy7ewwgnpmeabdmstm65hlgavqwmstm6dotjhuy6nupx.img
MD5: 2f282b84e7e608d5852449ed940bfc51
```



Thank You 😊

The End