Setup Instructions

These are generic instructions. If you want walkthrough instructions for ubuntu check this page.

You will need a django project in order to run SilverStrike. You can clone a sample application I recommend for production from here.

ch

verstrike.org

ike@googlegroups.com

oney using PayPal

This guide assumes you are cloning the project to `/srv/webapps/silverstrike`

To actually use it in production I recommend using uwsgi and nginx. If you have can contribute instructions for other servers, please do so! Below are two short config files for uwsgi and nginx that you can use to get your instance up and running. They are also included in the project you cloned so you can just symlink them to where they should be.

uwsgi.ini

```
[uwsgi]

chdir           = /srv/webapps/silverstrike
module          = wsgi
virtualenv      = /srv/webapps/silverstrike/env

master          = true
processes       = 10
socket          = /run/uwsgi/silverstrike.sock
#chmod-socket   = 666

vacuum          = true
plugin          = python3
```

```
uid             = www-data
gid             = www-data
```

nginx.config

```
server {
        listen 80;
        listen [::]:80;
        server_name silverstrike.example.com;
        return 301      https://$server_name$request_uri;
}

server {
        listen 443;
        listen [::]:443;
        server_name silverstrike.example.com;

        ssl                     on;
        ## More ssl settings here...

        location /static {
                alias /srv/webapps/silverstrike/public/static;
        }

        location / {
                uwsgi_pass  unix:/run/uwsgi/silverstrike.sock;
                include     /etc/nginx/uwsgi_params;
```

```
        }
}
```

Configuration

All configuration of SilverStrike happens in the `settings.py` file. You should be able to use it without modification, and you really don't need to know about every setting that is listed there, but here are a couple that you probably want to should change.

Name	Description
`SECRET_KEY`	Set it to a random string, this value is used internally by django and should be kept secret
`DEBUG`	setting it to `True` will allow you to know what's wrong if you're having difficulties, but you should set it to `False` once your instance is up and running. You might expose secret information if you set it to `True`

Name	Description
`ALLOWED_HOSTS`	Once you set `DEBUG` to `False` django will refuse to respond to requests that use a HOST header that's not included in this list. If you make Silverstrike available at `'https://silverstrike.example.com'` include that value in the list
`DATABASES`	By default sqlite is used, which is ok for development or small production setups, I recommend using a real database like postgresql or mariadb. Refer to the Django docs on how to configure the database. The link is provided in the settings file
`ACCOUNT_ADAPTER`	Since all users currently have access to all the data, signing up is disabled. If for some reason you want to enable signup comment the line of this setting.

Initialization

The development server is capable of serving all assets (stylesheets, images, javascript, ...) but is relatively slow in doing so. That's why in production (`DEBUG = False`) django doesn't serve them. So you will need to collect all static files and make them accessible to the web server. Here is the code to set everything up:

```
cd /srv/webapps/silverstrike
python3 -m venv env
source env/bin/activate
pip install silverstrike
python demo/manage.py migrate
python demo/manage.py collectstatic
python demo/manage.py createsuperuser
```

The migrate command initializes the database and applies any outstanding migrations. After updating to a new version you should run the command in case any new migrations have been added.

The collectstatic command collects all assets so that they can be served by the webserver directly. They are placed in `/srv/webapps/silverstrike/public/static/` . This command should be run everytime you update to a new version, because assets can always change.

The createsuperuser command can be used to create superusers with all permissions. After creating an initial superuser you can create more users using the builtin django admin.