Krypted Authentication Documentation

Release 3.0

Porowns, Zahren

Nov 02, 2020

Contents

CHAPTER 1

Overview

Krypted is a community management and authentication platform created by Krypted Gaming. *It is heavily influenced by EVE Online authentication platforms like SeAT and AllianceAuth.*

The purpose of Krypted is to help online gaming communities **manage their members**, **sync groups across a multitude of third-party platforms** (chat programs, voice programs, forums), and **provide useful tools for managing in-game entities** like EVE Online corporations, World of Warcraft guilds, and other popular MMO entities.

The Krypted platform is written in **Python**, utilizing the **Django** framework.

CHAPTER 2

Features

2.1 Core Features

2.1.1 Accounts

Accounts enables our login system, user profiles, and other crucial account features.

2.1.1.1 Enabling Accounts

1. Docker: Add `accounts` to the list of `EXTENSIONS` in the `.env` file
2. Other: Add `accounts` to the `EXTENSIONS` in the `settings.py` file

2.1.1.2 Notes

1. If there are no email settings (e.g `EMAIL_HOST`, etc), there will be no email verification.

2.1.2 Applications

Applications enables our application system, allowing users to submit applications for approval.

2.1.2.1 Enabling Applications

1. Docker: Add `applications` to the list of `EXTENSIONS` in the `.env` file
2. Other: Add `applications` to the `EXTENSIONS` in the `settings.py` file

2.1.2.2 Creating Templates

Templates are what users will fill out for submission, and will be created in your administration panel.

1. Navigate to your administration panel (https://DOMAIN/admin/)
2. Look for `Application templates`, this is where you will manage templates
3. Name and description are what you'd expect, this is what the user will see when deciding what to submit.
4. Questions are what questions you'd like people to answer on submission. Click the + icon to add questions, or select from existing.
5. "Groups to add" are groups that are added on application approval.
6. "Groups to remove" are groups that are removed on applcation approval or rejection.
7. "Required group" is a required group for a user to view this application template.

2.1.2.3 Applications

Under the sidebar menu, you'll see a new `Applications` tab. This is where you'll create and manage applications.

- The permission `View Application` is required to manage applications.
- The permission `Change Application` is required to approve or deny applications.
- If the `django_eveonline_connector` package is installed, applications will display the characters of the applicant.

2.1.3 Group Requests

Group Requests enables our group request system, which allows users to request groups.

This package adds a new `Groups` section to your sidebar, which is where you'll manage and request groups.

2.1.3.1 Open Groups

Open groups are a new option you'll see on your administration panel. These are groups that are automatically accepted.

2.1.3.2 Closed Groups

Closed groups are a new option you'll see on your administration pnael. These are groups hidden from users.

2.2 Krypted Packages

These are packages contributed and maintained by Krypted Gaming.

2.2.1 Installing Packages

2.2.1.1 Docker

If you're using Docker, you'll simply specify packages in the `.env` file.

1. Under the `REQUIREMENTS`, add KryptedGaming/<package_name> (e.g django-discord-connector).
2. Under the `EXTENSIONS`, add the extension name. This is what the package says to add into `INSTALLED_APPS`, e.g django_discord_connector).

2.2.1.2 PyPi

Typically, our packages are pushed to PyPi whenever we release.

1. Install the package `pip3 install <package_name>`
2. Follow the instructions in the package README.md
3. Add the package to your `settings.py` file (under `EXTENSIONS`, replace - with _)

2.2.1.3 Manually

If you want the latest version of our packages, you should install clone them and install them.

1. Clone the package `git clone <package_url>`
2. Install with pip `pip3 install -e <package_name>`
3. Follow instructions {2} and {3} from PyPi installation

2.3 Community Packages

These are packages contributed by the community.

2.3.1 Installing Packages

2.3.1.1 Docker

If you're using Docker, you'll simply specify packages in the `.env` file.

1. Under the `REQUIREMENTS`, add KryptedGaming/<package_name> (e.g django-discord-connector).
2. Under the `EXTENSIONS`, add the extension name. This is what the package says to add into `INSTALLED_APPS`, e.g django_discord_connector).

2.3.1.2 PyPi

Typically, our packages are pushed to PyPi when they are released.

1. Install the package `pip3 install <package_name>`
2. Follow the instructions in the package README.md
3. Add the package to your `settings.py` file (under `EXTENSIONS`, replace - with _)

2.3.1.3 Manually

If you want the latest version of a package, you should install clone them and install them.

1. Clone the package `git clone <package_url>`
2. Install with pip `pip3 install -e <package_name>`
3. Follow instructions {2} and {3} from PyPi installation

CHAPTER 3

Installation

3.1 Docker Installation

For production installation, we use Docker.

1. Grab our production-ready `docker-compose.yml` file (`wget https://raw.githubusercontent.com/KryptedGaming/krypted/development/conf/docker-compose.yml`)
2. Grab our example `.env` file (`wget https://raw.githubusercontent.com/KryptedGaming/krypted/development/conf/.env`)
3. Modify the `.env` with your desired settings
4. Launch your production environment with `docker-compose up -d`
5. By default, you'll be on port 8000. You'll need to proxypass this on your host webserver.

3.1.1 Environment Variables

Krypted supports numerous environment variables, which are set in your `.env` file.

3.1.1.1 Required Environment Variables

3.1.1.2 Optional Environment Variables

3.1.2 Example .env file

```
DJANGO_SECRET=secret
SITE_DOMAIN=localhost
SITE_TITLE=My Corporation
VERSION=development
```

(continues on next page)

(continued from previous page)

```
MYSQL_DATABASE=krypted
MYSQL_USER=krypted
MYSQL_PASSWORD=somepassword
MYSQL_HOST=db
MYSQL_PORT=3306
REQUIREMENTS=KryptedGaming/django-discord-connector,KryptedGaming/django-eveonline-
↪connector
EXTENSIONS=accounts,applications,group_requests,django_discord_connector,django_
↪eveonline_connector
```

We'll pick this apart piece by piece.

1. We set the `DJANGO_SECRET` to `secret`, which isn't very secure, but it's something. All of our hashing will be done by this, so if we ever need to migrate we need this value.
2. We set the `SITE_DOMAIN` to `localhost`, meaning users will access our website by going to `https://localhost`
3. We set the `SITE_NAME` to `My Corporation`, which will update certain areas of the site to have our name.
4. `MYSQL_DATABASE` This is the database name that we'll use.
5. `MYSQL_USER` This is the database user that has access to the above database.
6. `MYSQL_PASSWORD` We obviously need the password.
7. `MYSQL_HOST` Depending on where you're hosting your MYSQL database, you'll set this value. In this case, we're using docker-compose and our database is a local container on the same network.
8. `MYSQL_PORT` You'll likely not need to change this.
9. `REQUIREMENTS` Here's where the interesting part comes in. The REQUIREMENTS has two formats: `<Package Name>` and `<GitHub User>/<Repository Name>`. If we add something like `django_eveonline_connector`, pip will install it from the Python package index. If we specify `KryptedGaming/django-discord-connector`, we will grab it from GitHub.
10. Requirements aren't enough because some repositories will have different names. We might have a better solution in the future, but for now you need to specify all of the `EXTENSIONS` to be added to `INSTALLED_APPS`.

3.1.3 Using Packages

Each package has their own installation settings. Refer to that package for special instructions.

3.2 Local Installation

This is how to install a local copy of Krypted, which is primarily used by developers. **Do not use this method in production unless you're familar with making Django applications production ready.**

3.2.1 Clone the Repository

Clone the GitHub repository to your local file system. `git clone https://github.com/KryptedGaming/krypted.git`

3.2.2 Run the installation script

1. Navigate to the directory `cd krypted`
2. Ensure permissions are correct `chmod +x ./install/install.sh launcher`
3. Create a virtual environment `./launcher env`
4. Enter your virtual environment `source ./myenv/bin/activate`
5. Run the installation script `./launcher install`
6. Verify your installation `./launcher test`

3.2.3 Recommended: VSCode

We recommend you use VSCode when developing, because it's awesome.

1. Install VSCode from this link
2. Get the `Python` extension for VSCode
3. Get the `py-coverage-view` extension for VSCode (for code coverage when running `./launcher test`)

3.2.4 Recommended: Juniper Notebooks

By default, the developer requirements installs shell plus and notebook.

1. Enable `django-extensions` in your `INSTALLED_APPS`
2. Start notebook with `python3 manage.py shell_plus --notebook`
3. Create a notebook, start developing

By utilizing this tool, you'll be able to refresh your code changes and re-run previous code. It's a lifesaver for Python development, especially when debugging.

3.3 Docker Administration

This will go over how to administrate your container.

3.3.1 Basic Docker

To get things like `<container_name>` or other variables we may mention here, you'll need to know some basic Docker.

- `docker container ls` Lists all containers. You'll be able to get <container_name> from here.
- `docker-compose up` Starts up your containers from a `docker-compose.yml`. Run with `-d` to detach from terminal.
- `docker-compose down` Stops your containers from a `docker-compose.yml`
- `docker-compose build` If you're using our `Dockerfile`, this will rebuild the image.
- `docker-compose pull` If you're using our image, this will update your images to the latest.

3.3.2 Creating Superuser Accounts

Superuser accounts have all Django permissions, and are optimal for your admin accounts.

1. `docker container exec -it <container_name> python3 /opt/krypted/app/manage.py createsuperuser`
2. Fill out the information as required

3.3.3 Running Django Commands

Some packages require you to run setup scripts or other Django commands.

1. `docker container exec -it <container_name> python3 /opt/krypted/app/manage.py <command>`

3.3.4 Database Backups

Backups are important, you should do them frequently and every time before you updgrade.

1. Create a backup with

```
sudo docker-compose exec db sh -c 'exec mysqldump "$MYSQL_DATABASE" -u"$MYSQL_USER" -p
↪"$MYSQL_PASSWORD" 2>/dev/null' | gzip > "`date +"%Y_%m_%d"`_krypted.sql.gz"
```

1. Restore a backup with

```
zcat *krypted.sql.gz | docker-compose exec -T db sh -c 'exec mysql "$MYSQL_DATABASE" -
↪u"$MYSQL_USER" -p"$MYSQL_PASSWORD"'
```

CHAPTER 4

User Guide

4.1 Setup

4.1.1 Admin Panel

The administration panel is a feature of Django, and can be accessed by going to `https://<your_site>/admin`. This is where you set up package instances (e.g Discord and EVE Online), interact with the backend database, and create configuration for all Krypted packages.

To view the administration panel, a user must be `staff`, and have permissions to view each section. `superuser` accounts can view all sections, regardless of permissions.

4.1.2 Understanding Tasks

Most actions in the Krypted platform are Celery tasks, which allows us to perform backend actions without interrupting the end-user. There are two forms of tasks that we utilize: **Periodic Tasks** and **Signal Tasks**.

4.1.2.1 Signal Tasks

Signal tasks are tasks triggered by events in the web application. For example, if someone adds a new EVE Online character, we might want to re-verify their groups to make sure they are still eligible.

These are implemented by developers.

4.1.2.2 Periodic Tasks

These are tasks set by **YOU** to run in the background. For example, you might want to check every 5 minutes that everyones' Discord roles are up to date.

To create a Periodic Task:

1. Navigate to the Admin Panel

2. Click Periodic Tasks
3. Create a new task
4. Add a name for your task (e.g Update All Discord User Groups)
5. Select the registered task (e.g `sync_discord_user_groups`)
6. Add an Interval Schedule (e.g every 5 minutes)

Tasks will be your largest performance hit, so be smart about them.

4.2 Package Guides

4.2.1 Discord Connector

4.2.1.1 Overview

The Discord Connector allows you to attach Discord roles to Groups on the Krypted Platform. By doing this, users will automatically be assigned Discord roles whenever they are added to a Group on your website. In combination with other packages like EVE Group States and Group Requests, you're able to create a powerful syncing platform that enhances security across your platforms.

4.2.1.2 Quick Setup

To set up the `django_discord_connector` package:

1. Install it (varies, Docker vs Development)
2. (Recommended) Load the default schedule (`python3 manage.py loaddata discord_default_schedule`)
3. Create a Discord Application (https://discordapp.com/developers/applications)
4. Create a Discord client in the Admin Panel (`http://<your_site>/admin`)
5. Input the `CLIENT ID` from your Discord application
6. Input the `CLIENT_SECRET` from your Discord application
7. Click `OAuth2` on your Discord application, set your `CALLBACK URL` (e.g `http://<your_site>/discord/sso/callback`)
8. Click `Bot` on your Discord application, set your `TOKEN`
9. Create a non-expiry Discord invite link, input it
10. Invite your Bot to the Discord server (use the `OAuth2` bot tool)
11. Select your new client in the Admin Panel and click the dropdown, sync your groups

4.2.1.3 Detailed Setup

1. Create a Discord Application

1. Navigate to https://discordapp.com/developers/applications
2. Click "New Application"

3. Fill out a name, click create.
4. Add an application icon, description, and customize your application.
5. Copy the `CLIENT ID` value, save this for later.
6. Copy the `CLIENT SECRET` value, save this for later.

2. Get Bot Token

1. Click the **Bot** section
2. Add a Bot.
3. Customize your bot, add an icon and username.
4. Copy the `TOKEN` value, save this for later.

3. Inviting your Bot

1. Click the *OAuth2** section
2. Add a Redirect URL, `https://<your_domain>/discord/sso/callback` ((`your_site` is your site domain). Save this value for later.
3. Under the **Scopes** section, check the *bot* box.
4. Scroll down to **Bot Permissions**, check *Administrator*.
5. Scroll back up to **Scopes**, copy the URL at the bottom and paste it in your browser.
6. Use this URL to invite the bot to your server.

4. Get your Server ID

1. Open your Discord settings
2. Navigate to Appearance
3. Enable Developer Mode
4. Right click your Server icon in the Discord menu
5. Copy ID. Save this for later.

5. Create a permanant invite link

1. Navigate to your server, hover over a channel and create an invite link.
2. Select no expiry, unlimited uses. Save this link for later.

4. Creating the Discord Client

1. Navigate to your Admin Panel on the Krypted platform
2. Select Discord Clients
3. Create a new Discord client

4. Input the CALLBACK URL from Step #3
5. Input the SERVER ID from Step #4
6. Input the CLIENT ID from Step #1
7. Input the CLIENT SECRET from Step #1
8. Input the BOT TOKEN from Step #2
9. Input the INVITE LINK from Step #5

4.2.1.4 Tasks

There are a few tasks that you need to know about.

- `django_discord_connector.tasks.verify_all_discord_users_groups` is used to check that users have the groups they're supposed to, and updates their groups based on what's in the database. With the `DISCORD_REMOTE_PRIORITY` setting set to `True`, it will favor Discord groups over Krypted groups.
- `django_discord_connector.tasks.sync_discord_users_accounts` This will update usernames and nicknames of all users.
- `django_discord_connector.tasks.remote_sync_all_discord_users_groups` Sometimes, groups get out of sync. This is most common when someone adds a group to a user on the Discord server, instead of letting authentication handle it. By running this task, we do a hard sync on all users. **This is an expensive task, don't overuse it.**
- 'django_discord_connector.tasks.enforce_discord_nicknames`This task takes an ARGUMENT`enforce_strategy`, which can be on of the following:`EVE_ONLINE`. This will enforce Discord nicknames (e.g in the case of`EVE_ONLINE', primary character name). You must disable users' ability to change their nickname for this to work cleanly.

4.2.1.5 Recommended Task Schedule

4.2.2 EVE Connector

4.2.2.1 Overview

The EVE Online Connector is our base module for ESI. This adds characters, corporations, and alliance objects and the tasks to update them. *It's required by nearly other EVE Online package.*

4.2.2.2 Quick Steps

To set up the `django_eveonline_connector` package:

1. Install it (varies, Docker vs Development)
2. (Recommended) Load the default schedule (`python3 manage.py loaddata eveonline_default_schedule`)
3. Create a CCP Application (https://developers.eveonline.com/applications)
4. Add all scopes, enter your callback url (e.g `http://<your_site>/eveonline/sso/callback`)
5. Create an EVE Online client in the Admin Panel (`http://<your_site>/admin`)
6. Input the `CLIENT ID` from your application

7. Input the `CLIENT_SECRET` from your application

4.2.2.3 Detailed Steps

1. Create an EVE Online application

1. Navigate to https://developers.eveonline.com/applications
2. Log in, create a new application
3. Fill out a name. Recommended: [TICKER] - Krypted Platform
4. Fill out a description.
5. Under connection type, select Authentication & API Access.
6. Add all scopes.
7. Under callback URL, add `http://<your_site>/eveonline/sso/callback` (`your_site` is your site domain). Save this for later.
8. Save the application.
9. Copy the CLIENT ID. Save this for later.
10. Copy the CLIENT SECRET. Save this for later.

2. Create an EVE Online client

1. Navigate to your Admin Panel on the Krypted platform
2. Click EVE Clients
3. Create an EVE Client
4. Fill out the CALLBACK URL from Step #1
5. Fill out the CLIENT ID from Step #1
6. Fill out the CLIENT SECRET from Step #1

(Optional) Modifying Scopes

At any time, if you need to modify scopes, you can navigate to EVE SCOPES in the Admin Panel and create (or delete) scopes.

4.2.2.4 Tasks

- `django_eveonline_connector.tasks.update_all_characters` This will update all character tokens and their affiliations.
- `django_eveonline_connector.tasks.update_all_corporations` This will update all corporations and their affiliations.
- `django_eveonline_connector.tasks.update_all_alliances` This will update all alliances and their affiliations.

4.2.2.5 Permissions

4.2.2.6 Recommened Task Schedule

4.2.3 EVE Entity Extensions

4.2.3.1 Overview

The `django_eveonline_entity_extensions` module extends the EVE Connector, adding objects for journal entries, skills, transactions, contacts, contracts, and many other ESI entities.

It's great for light auditing and skill checks, and is required by more advanced auditing packages.

4.2.3.2 Quick Steps

1. Install `django_eveonline_entity_extensions`
2. See permissions and tasks

4.2.3.3 Permissions

All of the `view` permissions for EVE Online entity extensions are used. For example, if you want to allow someone to view assets, they need to have the view permission.

4.2.3.4 Tasks

Tasks for the entity extensions are fairly expensive, and make a ton of ESI calls. We recommend that you use them sparingly, and only update information that is needed.

Tasks are in the format `update_all_eve_<entity>_<action>`, For example, `update_all_eve_character_skills`.

4.2.4 EVE Group States

4.2.4.1 Overview

The `django_eveonline_group_states` package allows you to create group states that enable groups based on other Krypted groups, corporations, and alliances. For example, you may create a state called `Blue` which enables your alliance. Then, you would enable it so that members are that state are automatically added to a `Blue` group that syncs with Discord.

States are extremely powerful, and can be used in numerous ways to set up your community.

4.2.4.2 Quick Steps

1. Install `django_eveonline_group_states`
2. Navigate to EVE Group States in the Admin Panel
3. Create your group states

4.2.5 EVE Doctrine Manager

4.2.5.1 Overview

The doctrine manager allows you to add fittings, skill plans, and doctrines to your website. It also allows users to view these objects, and check their skills against fittings based on their skill plans.

4.2.5.2 Quick Steps

1. Install `django_eveonline_doctrine_manager`
2. Navigate to your Admin Panel
3. Create your doctrine types (e.g Shield, Armor) and fitting types (e.g DPS, Logistics)
4. Create fittings, doctrines, and skill plans

4.2.5.3 Permissions

Users will need permissions to view doctrines and fittings.

4.2.6 EVE Timerboard

4.2.6.1 Overview

The `django_eveonline_timerboard` allows you to create and track timers for EVE events.

4.2.6.2 Quick Steps

1. Install `django_eveonline_timerboard`
2. No additional setup required

4.2.6.3 Permissions

Users will need permissions to view timers.

CHAPTER 5

Developer Guide

5.1 Installation

Installing the Krypted platform developers is fairly simple, since we have scripts that will do most of the work.

5.1.1 Clone the Repository

Clone the GitHub repository to your local file system. `git clone https://github.com/KryptedGaming/krypted.git`

5.1.2 Run the installation script

1. Navigate to the directory `cd krypted`
2. Ensure permissions are correct `chmod +x ./install/install.sh launcher`
3. Create a virtual environment `./launcher env`
4. Enter your virtual environment `source ./myenv/bin/activate`
5. Run the installation script `./launcher install`
6. Verify your installation `./launcher test`

5.1.3 Recommended: VSCode

We recommend you use VSCode when developing, because it's awesome.

1. Install VSCode from this link
2. Get the `Python` extension for VSCode
3. Get the `py-coverage-view` extension for VSCode (for code coverage when running `./launcher test`)

5.1.4 Recommended: Juniper Notebooks

By default, the developer requirements installs shell plus and notebook.

1. Enable `django-extensions` in your `INSTALLED_APPS`
2. Start notebook with `python3 manage.py shell_plus --notebook`
3. Create a notebook, start developing

By utilizing this tool, you'll be able to refresh your code changes and re-run previous code. It's a lifesaver for Python development, especially when debugging.

5.2 Launcher

5.2.1 What is the launcher?

The launcher is our developer tool that handles a lot of the common commands that you'll be running. It's made to run in the root project directory, and helps you handle things like running Django, celery, and coverage. Understanding the launcher will save you a *ton* of time, and we highly recommend you use it when developing extensions.

5.2.2 Launcher Commands

5.2.3 Extending the Launcher

The launcher is written in Shell, so simply add a function and case statement for it, and you're good to go!

5.3 Running

During development, we typically just use the provided development tools by Django.

1. Ensure you're in the `app` directory `cd krypted/app`
2. Run the webserver `python3 manage.py runserver`
3. Navigate to `http://127.0.0.1:8000`

That's it! For production deployment, refer to the User Guide.

5.4 Contributing

Krypted accepts contributions in the form of **packages**, which are simply reusable Django Applications. All of our community packages can be found here.

Alternatively, you can contribute to our Krypted Packages in the form of pull requests.

There are a few requirements to become a community package,

1. The application must separate, with the preferred installation being `pip3 install <package>`
2. The application must have test cases, with decent coverage.
3. The application must be open source.

For mentorship on contributing, reach out to us on Discord.

CHAPTER 6

Upgrading

6.1 Upgrading to 4.2.x

Upgrading from 4.1.x to 4.2.x is fairly simple- there's not too many breaking changes.

6.1.1 Backup Database

Always back up your database before you mess around.

```
sudo docker-compose exec db sh -c 'exec mysqldump "$MYSQL_DATABASE" -u"$MYSQL_USER" -p
↪"$MYSQL_PASSWORD" 2>/dev/null' | gzip > "`date +"%Y_%m_%d"`_krypted.sql.gz"
```

6.1.2 Update Docker Image

To update, just updated your Docker image to `4.2.0` or `docker pull` the `latest` image.

6.1.3 Package Updates

Packages (and the platform) were updated to support new features. Below are the new minimum package versions.

Recommended Packages from our Alliance installation:

```
django-discord-connector==1.0.2
django-eveonline-timerboard==1.0.2
django-eveonline-group-states==1.0.0
django-eveonline-doctrine-manager==1.2.4
django-eveonline-connector==1.1.2
django-eveonline-buyback==0.0.1
```

6.1.4 Database Updates

In 4.2.x, we will be storing unicode data in the database. This means that you'll need to upgrade your MySQL tables (and database) to prevent errors.

```
https://gist.githubusercontent.com/porowns/d07e4e877cdf670e550e0fafb45fb6d6/raw/
↪52b6b618962335f28d019df413050262fc3858a2/4.2.x_migrate.sql
```

```
cat 4.2.x_migrate.sql | sudo docker-compose exec -T db sh -c 'exec mysql "$MYSQL_
↪DATABASE" -u"$MYSQL_USER" -p"$MYSQL_PASSWORD"'
```

Warning: This will delete all `django_eveonline_entity_extensions` data due to its deprecation.