Project Report

MOSS GitMate Milestone-I

Goals

- ➔ Extending all the plugins currently in the GitHub OAuth2 application to native GitHub app.
- ➔ Creating GitHub application authentication backend.

Learning Context & Procedure

Procedure

a. User-Server Authentication: OAuth2 Auth mechanism works the same with GitHub Apps and GitHub OAuth2 apps.
b. Server-Server Authentication: Requires new implementation of auto-refreshing JSON Web Tokens based on RSA public / private keys.
c. UI Changes: Requires documentation and modals for showing differences between GitHub Apps and GitHub OAuth2 applications.
e. Staged Changes: The GitHub App has been finalized and deployed here.

Technical Tradeoffs & Obstacles

As expected, everything went off good. We could ship all our deployments with proper permission scheme. Here is a list of issues, their solutions and trade-offs during the implementation.

a. Frontend of the application had an outdated **@angular/material** library with no available documentation as of today.
b. Modifications in the Repository model, due to use of unique identifiers instead of full names of the repositories. (which required a database migration in production)
c. Use of JSON Web Tokens brought about a few problems during unit-tests, which were easily solved by using a randomly generated private key.
d. OAuth2 authorization issue with python-social-auth for GitHub apps. (solved by making our own implementation of **GitHubAppOAuth2** class)

e. The GitHub API endpoint to list repositories connected to a user is inaccessible to User-Server authentication of GitHub apps. So, we had to trade off the API and depend upon GitHub's own app management system for addition and deletion of repositories.

Commits involving the update (in the order of their merge)

Changes from GitMate-2 repository:

- Return 406 for installation repo activation
- gitmate_hooks/responders: Delete OAuth webhook
- API Documentation: Use restframework built-in docs
- 0019_change_pk_to_identifier: Make it reversible
- RecordedTestCase: Create class to record requests
- Installation: Set default identifier to None
- models: Allow null identifier for repositories
- Remove inactive repos without identifiers
- Repository: Set installation to null when removed
- Repository.token: Return installation token
- settings: Add GitHub provider for social auth
- config.views: Add installation repos to listing
- Add pipeline to populate repositories
- gitmate/backends.py: Add GitHubAppOAuth2 backend
- Installation: Add admins ManyToMany field

Changes from IGitt repository:

- Return sender from GitHub installation webhook
- User: Add get_installations method
- GitHub.handle_webhook: Return installed repos
- GitHub: Add support for installation events
- GitHubJsonWebToken: Add minor correction to expiry
- User: Add installed_repositories method
- Installation: Add Installation interface
- GitHubInstallationToken: Add jwt property
- GitHubJsonWebToken: Use current timezone
- Add GitHubInstallationToken
- Add GitHubJsonWebToken
- requirements: Add cryptograpy, PyJWT

Project Report

MOSS GitMate Milestone-II

Goals

- ➔ Extensible caching for outbound GET requests in IGitt with compatibility for most python frameworks.
- ➔ Utilizing the caching mechanism in gitmate-2 with a database backed cache for persistence.

Learning Context & Procedure

Procedure

a. Implementing a low level in-memory cache within IGitt by default.
b. Designing a singleton Cache class that interacts with get and set methods to retrieve and update the cache respectively.
c. Unittesting that the Cache class works as expected.
d. Documenting the use cases of Cache. The documentation is available here.
e. Utilising the cache within GitMate with a database backend to elicit persistence. Django has built-in support for different cache backends.

Technical Tradeoffs & Obstacles

Things were easy here and went better than expected. We could roll out the library with built-in support for many frameworks like Django, Flask, etc. There were absolutely no trade-offs in implementing the changes here. The only obstacle we ever faced was deciding on how to invalidate the cache. We tackled it by using GitHub API for conditional requests, which combines the use of an E-Tag header that can be used to fetch new data / use the existing cache based on the modifications to the resource. GitLab also provides a similar approach for conditional requests.

Commits involving the update (in the order of their merge)

Changes from GitMate-2 repository:

- Override IGitt cache with database backend

Changes from IGitt repository:

- docs: Using IGitt cache
- GitHub: Store webhook data to cache
- Use Cache for storing requests
- Utils: Add Cache class to manage cache