django-otp-agents Documentation

Release 0.3.0

Peter Sagerson

Jul 22, 2017

Contents

Package Documentation

This uses django-agent-trust to add OTP-based machine authorization to django-otp.

See django-otp for more information on the OTP framework.

This version is supported on Python 2.7 and 3.4+; and Django 1.8 and 1.10+.

CHAPTER 1

Installation

This is technically a Django app, but only for testing purposes. To use this project, just make sure you have django-otp and django-agent-trust installed correctly.

CHAPTER 2

Forms

This package provides subclasses of `OTPAuthenticationForm` and `OTPTokenForm` with an extra boolean field called `otp_trust_agent`. The user can check this option to indicate that they'd like to bypass OTP verification when accessing the site from the same browser in the future.

When these forms are used with `django.contrib.auth.views.login()`, we will call the appropriate django-agent-trust APIs automatically. If the trust option is selected, the agent will be trusted persistently; if not, it will be trusted for the current session. Views willing to accept a trusted agent in lieu of OTP verification may then use `trusted_agent_required()` in place of `otp_required()`.

class otp_agents.forms.**OTPAuthenticationForm**(*request=None*, **args*, ***kwargs*)
Extends `OTPAuthenticationForm` with support for agent trust.

class otp_agents.forms.**OTPTokenForm**(*user*, *request=None*, **args*, ***kwargs*)
Extends `OTPTokenForm` with support for agent trust.

CHAPTER 3

Views

otp_agents.views.**login**(*request*, ***kwargs*)

This is just like `django_otp.views.login()` except that it uses our agent-trust-enabled forms.

CHAPTER 4

Decorators

If you use any OTP-aware authentication forms that are *not* trusted-agent-aware, then the following decorator may be useful. For example, suppose you have one view protected by `otp_required()` and another that is protected by `trusted_agent_required()`. If an authenticated user visits the first view, he will be asked to provide an OTP token via the standard `OTPTokenForm`, which will verify the user, but will not set up a trusted agent. If he then visits the second view, he will be asked to verify again, this time with the trusted-agent-aware *`OTPTokenForm`*. This is clearly not what you intend, since OTP verification should be more than sufficient to authorize the user for the second view.

To get around this, the following decorator effectively merges `otp_required()` and `trusted_agent_required()` into a single decorator that will be satisfied with either a verified user or a trusted agent. The default behavior is to act exactly like `otp_required()`; pass `accept_trusted_agent=True` to enable the more lenient policy.

`otp_agents.decorators.`**`otp_required`**(*view=None*, *redirect_field_name='next'*, *login_url=None*, *if_configured=False*, *accept_trusted_agent=False*)

Similar to `otp_required()`, but with an extra argument.

The default value for `login_url` depends on the value of `accept_trusted_agent`. If `True`, we'll use `AGENT_LOGIN_URL`; otherwise, we'll use `OTP_LOGIN_URL`.

Parameters **`accept_trusted_agent`** (*`bool`*) – If `True`, we'll accept a trusted agent in lieu of OTP verification. Default is `False`.

CHAPTER 5

Admin

class otp_agents.admin.**TrustedAgentAdminSite**(*name='otpadmin'*)

This is an OTPAdminSite subclass that is satisfied by a trusted agent.

This is identical to django_otp.admin.OTPAdminSite except that it will skip the login dialog if the user is already logged in and is either verified with an OTP device or is coming from a previously trusted agent.

Note that AdminSite lacks the hooks necessary to properly implement a trusted agent login form here. If the user does not meet the criteria above, they will have to verify themselves with an OTP device to log in. The experience will be better if you send users through your own custom login flow before redirecting them to this admin site.

CHAPTER 6

Changes

changes

CHAPTER 7

License

Python Module Index

o

Index

L

O

T