Notes on the IP address-based redirection middleware

Srijan Technologies Pvt. Ltd., India
http://www.srijan.in

Document version: 1.0
July 6, 2009

Contents

Chapter 1

Introduction

This document describes the IP address redirection solution built for the Ciboe image database. It fulfils a need to have users from a set of registered IP addresses provided access to non-admin views without their having to log in. Administrative usrs from these IP addresses, namely anyone wishing to view a Django admin. page will need to log in as usual. For users from any unregistered IP addresses, the authentication process is as usual for Django, namely access is allowed if the user is authenticated and has sufficient permissions for the view in question.

The solution is implemented as Django middleware, also necessitating some changes to the URLs, and views for the Django project. Chapter 2 describes installation and usage, and should be all that is required for a user of the middleware. Chapter 3 describes how the users of the system interact with it. Some technical details are provided in Chapter 4. Finally, policy decisions, edge cases, and peculiarities are described in Chapter 5.

Chapter 2

Installation and usage

2.1 External dependencies

The middleware uses the `netaddr` [3] library to manipulate IP addresses. `netaddr` was chosen after a careful examination of available Python IP libraries. The other strong contender was `ipaddr` [2] which has been accepted into the standard library for Python versions 2.7 and 3.1 onwards. The final choice of `netaddr` was because of better functionality, more features, and especially the end-user friendly classes like IPRange class for IPAddress ranges, and the Wildcard class. We use only a tiny part of the functionality provided by `netaddr`, and a conscious decision was made not to offer too many configuration options. However, if it is felt to be desirable, features like wildcards for IP addresses can be added. The current version of the library in use is 0.6.3.

2.2 Changes to `settings.py`

The following changes will need to be made to `settings.py` in order to use this middleware. All changes are actually made in `settings_local.py`:

- *Settings for middleware classes:* The IPAddressMiddleware resides in the file `middleware.py`, and can be added to the MIDDLEWARE_CLASSES variable in `settings.py` as 'middleware.IPAddressMiddleware'. The IPAddressMiddleWare can be added any where in the list of middleware classes. Thus, the MIDDLEWARE_CLASSES variable could be set like:

```
MIDDLEWARE_CLASSES = (
  'django.middleware.common.CommonMiddleware',
  'django.contrib.sessions.middleware.SessionMiddleware',
  'django.contrib.auth.middleware.AuthenticationMiddleware',
  'middleware.IPAddressMiddleware',
)
```

- *Specifying registered IPs:* This is given in the variable SRJ_IP_REG as a list, where each entry can be either a single IP address value specified as a string, or a sequence of two values as strings, which specify the start and end of an IP address range. Thus,

```
SRJ_IP_REG = [
    '192.168.10.1', '192.168.10.4', ['192.168.10.6', '192.168.10.24'],
    '192.168.10.40'
    ]
```

Enhancements are also possible, e.g., the individual values can also themselves be ranges, e.g., '192.168.10.0/3'. For details, see the examples in `http://code.google.com/p/netaddr/wiki/IPv4Examples`. Still further extensions are possible, e.g., see `http://code.google.com/p/netaddr/wiki/WildcardExamples`, and the complete API at `http://packages.python.org/netaddr/`. However, in the interests of simplicity, these are not currently included.

- *Templates for login pages:* There are separate login pages for users from registered, and unregistered IPs. These are configured through the variables SRJ_TMPL_LOGIN_REG and SRJ_TMPL_LOGIN_UNREG, e.g.,

```
SRJ_TMPL_LOGIN_REG = 'login_reg.html'
SRJ_TMPL_LOGIN_UNREG = 'login.html'
```

Please note that these are set in the middleware, and hence specifying templates as `template_name` arguments to the dictionary for the `login` view will not work.

2.3 Running

Chapter 3

Functioning of the middleware

3.1 Registered IP addresses

3.2 Unregistered IP addresses

Chapter 4

Technical details

Chapter 5

Issues

Bibliography

[1] A wiki write-up comparing netaddr to other Python IP address manipulation libraries. Though this is from the netaddr site, and might be biased, it does offer an argument for netaddr. `http://code.google.com/p/netaddr/wiki/YetAnotherPythonIPModule`.

[2] ipaddr is an IP maipulation library in Python. `http://code.google.com/p/ipaddr-py/`.

[3] netaddr is a Python library for the manipulation of various common network address notations and representations. `http://code.google.com/p/netaddr/`. netaddr is a super-set of the functionality in most of the Python IP address modules available on the net. See the Wiki note [1].