factored Documentation

Release 3.0.1

Nathan Van Gheem

Jul 26, 2017

Contents

Factored is a comprehensive 2-factor authentication system that works with any web technology.

Contents:

CHAPTER 1

Introduction

Factored is a comprehensive 2-factor authentication system that works with any web technology.

Authorization Types

Factored uses a plugin system to provide different types of authorizations. Out of the box, it provides Google Authenticator and Email Token support.

Integration Strategies

Factored supports different types of integrations for your web applications.

- Proxy: factor is in front of web application and sends authorized requests to configured web application.
- Web server plugins: Nginx and ATS plugins are avilable using lua.
- WSGI: A filter is provided to be use with Python Web Applications that utilize this standard.

CHAPTER 2

Installation

Download

Choose a release from here or just download the latest:

```
wget https://github.com/wildcardcorp/factored/archive/master.zip
unzip factored-master
mv factored-master factored
```

Or clone git for latest

from github:

```
git clone git@github.com:wildcardcorp/factored.git
```

build

The project uses buildout to manage dependencies:

```
cd factored
virtualenv .
./bin/python bootstrap.py
./bin/buildout
```

Test

with default sample config:

```
./bin/factored_initializedb develop.ini
./bin/pserve develop.ini
```

You should now have a factored server running on port 8000.

Using proxy

Install WSGIProxy:

```
./bin/easy_install WSGIProxy
```

CHAPTER 3

Configuration

Must follow the example develop.ini provided. You'll probably want to copy that file into your own and change the settings.

Edit server and port settings for application server if not using with another wsgi application.

INI configuration options

appname Appened to google auth keys so it doesn't overwrite others.

auth_tkt.secret The secret (a string) used for auth_tkt cookie signing.

auth_tkt.cookie_name The cookie name used

auth_tkt.secure Only send the cookie back over a secure conn.

auth_tkt.include_ip Make the requesting IP address part of the authentication data in the cookie.

auth_tkt.path The path for which the auth_tkt cookie is valid. May be desirable if the application only serves part of a domain.

auth_tkt.http_only Hide cookie from JavaScript by setting the HttpOnly flag. Not honored by all browsers.

auth_tkt.wild_domain An auth_tkt cookie will be generated for the wildcard domain.

auth_tkt.cookie_domain To set the cookie on an additional different domain.

auth_tkt.hashalg Set to use a different hashing algorithm. Defaults to sha512

auth_timeout The amount of time in seconds a normal authentication session is valid for.

auth_remember_timeout The amount of time in seconds the authentication seesion is valid for when the user clicks "remember me."

base_auth_url Base url all authentication urls and resources are based off of. Useful if you're only looking to authenticate a portion of a site.

excepted_paths the path part of a request URL is compared to items in this list (one per line) and if there is a match, the request will be processed normally without any factored authorization.

supported_auth_schemes Supported authentication schemes.

email_auth_window If using email authentication, the window of time the user has to enter correct code in.

em.subject Email authencation subject used.

em.sender Email authentication from address.

em.body Email Authentication text body. *{code}* will be replaced with the code. *{url}* will be replaced with a URL that users can use to perform the authentication directly, without entering the *{code}* on the form.

em.salt Salt value used to generate parts of the *{url}* value.

em.url_remember Accepts 'True' or 'False' values. If 'True' then the generated *{url}* will indicate the auth should be remembered for the *auth_remember_timeout* instead of the *auth_timeout*

sms.auth_window number of seconds before an sms authentication code expires

sms.userlist url to a JSON endpoint for getting a phone number for a particular user.

sms.msg_tmpl sms message template, must have {code} in the body of the text.

sms.plivo_auth_id plivo auth_id to use with plivo api

sms.plivo_auth_token plivo auth_token to use with plivo api

sms.plivo_phone_number source phone number to use with plivo api

pyramid. prefixed options Configuration passed directly into pyramid configuration.

sqlalchemy.url Connection string for sql backend. Most configurations will work fine with normal sqlite.

mail. prefixed options Configuration passed directly to the mailer plugin. Options can be found at http://packages.python.org/pyramid_mailer/#configuration

autouserfinder Specify a plugin that will automatically find users for the system to allow authentication for. Prepackaged plugins include *SQL* and *Email Domain*.

allowcodereminder (truelfalse) value defaulting to false that allows the user, if the username is an email, to get a reminder of their code sent to them.

allowcodereminder.subject If using allowing code reminders, the email subject of reminder

allowcodereminder.sender If using allowing code reminders, the email from address of reminder

allowcodereminder.body If using allowing code reminders, the email body of reminder

formtext.title customize title of form

formtext.legend customize legend of form

formtext.username.label customize username label

formtext.username.desc customize username description

formtext.code.label customize code label

formtext.code.desc customize code description

formtext.button.username customize username buttion text

formtext.button.authenticate customize athenticate buttion text

formtext.button.codereminder customize code reminder buttion text

formtext.error.invalid_username_code customize invalid username code text

formtext.error.invalid_code customize invalid code text

formtext.error.invalid_username customize invalid username text

formtext.error.code_reminder customize invalid code reminder text

autouserfinder SQL configuration options

autouserfinder.connection_string sqlalchemy connection string to connection to the database.

autouserfinder.table_name Name of the table to lookup users in.

autouserfinder.email_field Name of the field to find the usernames(could be username or email field).

autouserfinder Email Domain configuration options

autouserfinder.valid_domains List of valid domains to automatically create users for.

autouserfinder LDAP configuration options

autouserfinder.conn_string LDAP connection string, IE 'ldaps://127.0.0.1:636'

autouserfinder.check_certificate if [true] then the LDAPS SSL certificate is checked for validity

autouserfinder.starttls if [true] then STARTTLS is attempted

autouserfinder.bind_dn DN of the bind user, IE 'cn=binduser,cn=Users,dc=example,dc=com'

autouserfinder.bind_pw Password for the Bind DN

autouserfinder.base_dn Base DN to perform search for users on, IE 'cn=Users,dc=example,dc=com'

autouserfinder.username_attr This will typically be 'mail' or another attribute that stores an email address for the user

autouserfinder.lookup_timeout Number of seconds before an LDAP search will timeout

Nginx Example Configuration with proxying

An example setup with nginx and load balancing:

```
server {
    listen  80;
    server_name www.test.com;
    include proxy.conf;

    # paths to protect
    location ~ ^/admin.* {
        proxy_pass http://127.0.0.1:8000;
    }

    location / {
        proxy_pass http://app;
    }
```

```
}

server {
    listen 8090;
    include proxy.conf;
    location / {
        proxy_pass http://app;
    }
}
```

Then factored would be configured to run on port 8000 and proxy to 8090 and have *base_auth_url* url set to /admin/auth.

Sample INI Configuration

An example to follow if you're not using a git checkout:

```
[app:proxy]
use = egg:factored#simpleproxy
server = 127.0.0.1
port = 8090
urlscheme = http

[filter-app:main]
use = egg:factored#main
next = proxy
appname = REPLACEME

auth_tkt.secret = REPLACEME
auth_tkt.cookie_name = factored
auth_tkt.secure = false
auth_tkt.include_ip = true

auth_timeout = 7200
auth_remember_timeout = 604800
base_auth_url = /auth
supported_auth_schemes =
    Google Auth
    Email

email_auth_window = 120
# in seconds
em.subject = Authentication Request
em.sender = foo@bar.com
em.body =
    You have requested authentication.
    Your temporary access code is: {code}

autouserfinder = SQL
autouserfinder.table_name = users
autouserfinder.email_field = email
autouserfinder.connection_string = sqlite:///%(here)s/users.db

allowcodereminder = true
allowcodereminder.subject = Authentication code reminder
allowcodereminder.sender = foo@bar.com
```

```
allowcodereminder.body =
    You have requested code reminder.
    Your google auth code url is: {code}

pyramid.reload_templates = true
pyramid.debug_authorization = true
pyramid.debug_notfound = true
pyramid.debug_routematch = true
pyramid.default_locale_name = en
pyramid.includes =
    pyramid_tm
    pyramid_mailer

sqlalchemy.url = sqlite:///%(here)s/test.db

# all mail settings can be found at http://packages.python.org/pyramid_mailer/
↪#configuration
mail.host = localhost
mail.port = 25

[server:main]
use = egg:waitress#main
# Change to 0.0.0.0 to make public:
host = 127.0.0.1
port = 8000
```

With Gunicorn

Install:

```
./bin/easy_install gunicorn
```

to run:

```
./bin/gunicorn_paste --workers=2 develop.ini
```

Customizing Templates

Use pcreate to generate package skeleton:

> ./bin/pcreate –template=starter factored_customize cd factored_customize

To register template overrides customize __init__.py:

> from factored.templates import registerTemplateCustomizations
>
> **def includeme(config):** import factored_customize as pkg registerTemplateCustomizations(config, 'templates', pkg)

Modify paster ini file to include pyramid addon:

> **pyramid.includes =** ... factored_customize

Available Customizable Templates

meta.pt Override metadata in the head tag.

includes.pt Override includes in the head tag.

headbottom.pt Add additional html to the bottom of the head tag. Empty by default.

top.pt Renders at top of container. Empty by default.

title.pt Renders title of application.

abovecontent.pt Renders above content. Empty by default.

auth.pt Authentication layout template.

auth-code.pt Code input.

auth-email.pt Email input.

auth-controls.pt Form controls.

auth-chooser.pt Authentication system chooser.

belowcontent.pt Below the content. Empty by default.

footer.pt Application footer.

bottom.pt Very bottom of layout. Empty by default.

CHAPTER 4

User Management

If you do not use an auto user plugin and/or you need to configure a users google auth plugin, pay attention to the commands below.

Make sure you have first installed and initialized the database.

- Add user:

```
./bin/factored_adduser develop.ini --username=foo@bar.com
```

- Delete user:

```
./bin/factored_removeuser develop.ini --username=foo@bar.com
```

- List user info:

```
./bin/factored_listuserinfo develop.ini --username=foo@bar.com
```

- List users:

```
./bin/factored_listusers develop.ini
```

CHAPTER 5

Web Server Plugins

factored offers to web server plugins: nginx and apache traffic server. Both plugins require the web server to be compile with lua support.

Nginx

Requirements

- lua 5.1 and dev packages
- nginx compiled with http://wiki.nginx.org/HttpLuaModule
- requires using sha256 auth_tkt hash algorithm

Example Config

You'll need to customize this:

```
location / {
      # Thes must match factored config
      set $fcookie_name 'pnutbtr';
      set $fsecret 'secret';
      set $finclude_ip 0;
      set $ftimeout 0;

      set $authenticated 0;
      set $path '';
      set $proxyto '127.0.0.1:8000/';
      set_by_lua_file $authenticated /path/to/installed/factored/plugins/nginx.lua;
      if ($authenticated = 1) {
        set $proxyto '127.0.0.1:8080/';
        set $path 'VirtualHostBase/http/www.foobar.com:80/Plone/VirtualHostRoot/';
      }
```

```
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_pass http://$proxyto/$path$request_uri;
    }
```

Apache Traffic Server (ATS) Plugin

Requirements

- ATS 7.2.x (tested on 7.2.0) configured as a reverse proxy server
- ATS Lua plugin

See the Apache Traffic Server Documentation for how to install and configure it. Take note that you will need to use a version of ATS compiled with the *–enable-experimental-plugins*, or you will need to configure your installation to work with the TS-Lua plugin.

Install

Put the following into a file that is readable by ATS (ex: `/etc/factored/plugin.lua`)::

```
--
--  These should match your factored settings (IE the values in your INI
--  configuration file). This value SHOULD be called "factored_settings".
--
factored_settings = {
  -- the HOST and PORT Factored is running on
  scheme='http',
  host='127.0.0.1',
  port=8000,

  -- AUTH TKT settings
  cookie_name='your_auth_tkt_cookie_name',
  secret='your_auth_tkt_secret_here',
  include_ip=false, -- [true] to include IP in cookie value
  timeout=false, -- [true] to manually handle cookie timeouts

  -- PLUGIN directory -- by default factored has a "plugins" directory
  -- which contains several lua files that are necessary. This directory
  -- should contain "ats.lua", "factored.lua", "bit.lua", and "sha.lua"
  basepath='/path/to/factored/plugins/'
}

--------------------------------------------------------------------------------
-- ## PAST THIS POINT YOU SHOULDN'T NEED TO MODIFY ############################
-- (but it is required)
--
require 'package'
if string.find(package.path, factored_settings.basepath) == nil then
    ts.add_package_path(factored_settings.basepath .. '?.lua')
end
ats = require 'ats'
```

```
function do_remap()
  ts.http.set_debug(0)
  local status, ret = pcall(ats.do_remap)
  -- if the pcall was successful, then we should be able to return
  -- the result of the pcall
  if status then
    return ret
  else
    -- this is a special case, if something went wrong in the normal
    -- remap process, the url will be intercepted with a 403 message
    -- if you want a customized message, put your own intercept function here
    ts.http.intercept(ats.factored_failed)
    return 0
  end
end
```

Then in your ATS `remap.config` file, you'll want a line like the following::

```
map TARGET REPLACEMENT @plugin=/path/to/tslua.so @pparam=/path/to/your/custom/
↪settings.lua
```

Where 'TARGET' would be the incoming URL and 'REPLACEMENT' is the upstream (NOT the factored server, but whichever URL you want behind factored).

The `/path/to/tslua.so` is going to be based on your installation – a default ATS installation from source on Ubuntu will put it in `/usr/local/libexec/tslua.so`. Note – the full path is necessary.

The `/path/to/your/custom/settings.lua` would be the path to the file that contains your customized factored configuration (`/etc/factored/plugin.lua` from the example above). Note – the full path is necessary.

This plugin works by checking the auth_tkt cookie on each request – if there is a cookie, and it's valid, then the plugin just passes on factored entirely, letting ATS continue with the request process. If the cookie is not found or not valid, the plugin will re-write the upstream to point at the configured factored server.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search