链滴

简易版配置文件管理实现

作者：evling

原文链接：https://ld246.com/article/1642683356859

来源网站：链滴

导读

配置、证书、密钥文件这类经常会涉及到变更，如果每次都到登录到目标服务器进行操作，要是量特大，着实不美丽，那这里小易给大家伙带来一个简易的实现方案，供各位参考。

需求分析

配置、密钥文件往往存储了比较机密的信息，安全性要保证它不被轻易的泄漏，第一首要条件必须加证，标题都说了简易版，那直接上个基本认证够用了，在加上传输层的安全性，那就基本认证配合https足够。那么咱们再抛一个问题，用一个账号进行访问所有的文件？显然安全风险还是太高，毕竟，了便于记忆，咱们有几个是把文件名设置成随机码的对吧，所以还得细划下访问控制，暂定一个账号问一小批指定的文件吧，咱可以这样定义，某一台机器的所有相关变更的文件给它一个账号统一访问具体到某台机器上咱就设置个定时任务，拿着这个账号下载对应的文件与原文件做比对，有变更就替之。

概要设计

咱再来捋一捋，涉及文件变更，得有文件管理吧，涉及多账号，得有账号管理吧，具体某个账号还不访问它不该访问，那给它设置个访问清单得有吧，所以，思路明晰了。

文件管理

- 文件上传，通过Web页面，或者直接API调用，上传限制大小，后缀名白名单，防止上传至任意目破坏系统文件
- 文件下载，提供一个接口供消费客户端获取配置文件
- 文件删除，暂不考虑

账号管理

- 用户新建，用户已存在的，忽略操作
- 认证凭证保护，密码不能明文存放，这里仅用于账号认证，采用单向hash
- 用户删除，暂不考虑

访问控制

- 设置两个角色，一个管理员，若干个普通用户，管理员可可进行文件上传、账号管理及设置访问清，普通用户仅能访问该访问的文件
- 提供一个接口，供管理员设置普通用户的文件访问清单

代码实现

服务端

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# @Project : tools
# @File : conf_server.py
# @Software: PyCharm
```

```
# @Author : 易雾君
# @Email : evling2020@gmail.com
# @公众号 : 易雾山庄
# @Site : https://www.evling.tech
# @Describe : 家庭基建，生活乐享.
# @Time : 2022/1/17 9:38 PM
import json
import os
from flask import Flask, flash, request, redirect, url_for, send_from_directory, jsonify
from flask_httpauth import HTTPBasicAuth
from werkzeug.utils import secure_filename
from werkzeug.security import generate_password_hash, check_password_hash

UPLOAD_FOLDER = './uploads'
DATA_JSON = './data.json'
ALLOWED_EXTENSIONS = {'txt', 'conf', 'sh', 'crt', 'key', 'json'}

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1000 * 1000     # 限制文件大小为 16M
app.add_url_rule(
    "/uploads/<name>", endpoint="download_file", build_only=True
)
auth = HTTPBasicAuth()


users = {
    'admin': {
        'password': generate_password_hash('admin'),
        'privileged': True,
        'files': []
    },
}

def save_data():
    with open(DATA_JSON, 'w', encoding='utf-8') as f:
        json.dump(users, f, ensure_ascii=False, indent=4)
        f.close()

if not os.path.exists(UPLOAD_FOLDER):
    os.mkdir(UPLOAD_FOLDER)

if os.path.exists(DATA_JSON):
    with open(DATA_JSON, 'r') as f:
        users = json.load(f)
        f.close()
else:
    save_data()

def is_admin(username):
    return users.get(username).get('privileged')

def can_access(username, filename):
    return True if filename in users.get(username).get('files') else False
```

```
def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@auth.verify_password
def verify_password(username, password):
    if username in users and check_password_hash(users.get(username).get('password'), passw
rd):
        return username

@app.route('/users/add', methods=['POST'])
@auth.login_required
def add_user():
    if not is_admin(auth.current_user()):
        return jsonify({'msg': 'You can not access this page!'})
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username not in users:
            users[username] = {'password': generate_password_hash(password), 'privileged': False,
files': []}
            save_data()
            return jsonify({'msg': 'Add user succ'})
        else:
            return jsonify({'msg': 'User: {} may be exist'.format(username)})

@app.route('/users/passwd', methods=['POST'])
@auth.login_required
def passwd():
    if not is_admin(auth.current_user()):
        return jsonify({'msg': 'You can not access this page!'})
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if username in users:
            users[username]['password'] = generate_password_hash(password)
            save_data()
            return jsonify({'msg': 'Change {}\'s password succ'.format(username)})
        else:
            return jsonify({'msg': 'User: {} not exist!'.format(username)})

@app.route('/users/authorize', methods=['POST'])
@auth.login_required
def authorize():
    if not is_admin(auth.current_user()):
        return jsonify({'msg': 'You can not access this page!'})
    if request.method == 'POST':
        data = request.get_json()
        username = data['username']
        files = data['files']
        if username in users:
            succ_list = []
            fail_list = []
            for file in files:
```

```
            if not os.path.exists(os.path.join(UPLOAD_FOLDER, file)):
                fail_list.append(file)
            else:
                succ_list.append(file)
        if succ_list != []:
            for file in succ_list:
                if file not in users[username]['files']:
                    users[username]['files'].append(file)
            save_data()
            return jsonify({'msg': 'authorize {} succ'.format(username), 'succ_list': succ_list, 'fail_li
t': fail_list})
        else:
            return jsonify({'msg': 'No file authorized to {}'.format(username), 'succ_list': succ_list,
fail_list': fail_list})
    else:
        return jsonify({'msg': 'User: {} not exist!'.format(username)})

@app.route('/uploads/<path:filename>', methods=['GET'])
@auth.login_required
def download_file(filename):
    if can_access(auth.current_user(), filename):
        return send_from_directory(app.config["UPLOAD_FOLDER"], filename)
    else:
        return jsonify({'msg': 'You can not do this operate!'})

@app.route('/', methods=['GET', 'POST'])
@auth.login_required
def upload_file():
    if not is_admin(auth.current_user()):
        return jsonify({'msg': 'You can not access this page!'})
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # If the user does not select a file, the browser submits an
        # empty file without a filename.
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            return redirect(url_for('download_file', name=filename))
    return '''
    <!doctype html>
    <title>Upload new File</title>
    <h1>Upload new File</h1>
    <form method=post enctype=multipart/form-data>
      <input type=file name=file>
      <input type=submit value=Upload>
    </form>
    '''
```

```
if __name__ == '__main__':
    app.run(port=5001)
```

API梳理

- /users/add

```
# 新建用户，仅管理员有权限访问
auth=('admin', 'admin')    # 默认密码为admin
res = requests.post(url=base_url + "users/add", data={'username': 'your_username', 'password
: 'your_password'}, auth=auth)
print(res.status_code)
print(res.json())
```

- /users/passwd

```
# 修改用户密码，仅管理员有权限访问
auth=('admin', 'admin')    # 默认密码为admin
res=requests.post(url=base_url+"users/passwd",data=post_data,auth=auth)
print(res.status_code)
print(res.json())
```

- /users/authorize

```
# 授权文件访问，仅管理员有权限访问
auth=('admin', 'admin')    # 默认密码为admin
res = requests.post(url=base_url + "users/authorize", json={'username': 'your_username', 'files
: 'your_files_list'}, auth=auth)
print(res.status_code)
print(res.json())
```

- /

```
# 文件上传，仅管理员有权限访问
auth=('admin', 'admin')    # 默认密码为admin
res = requests.post(base_url, files={'file': open('example.txt', 'rb')} ,auth=auth)
print(res.content)
```

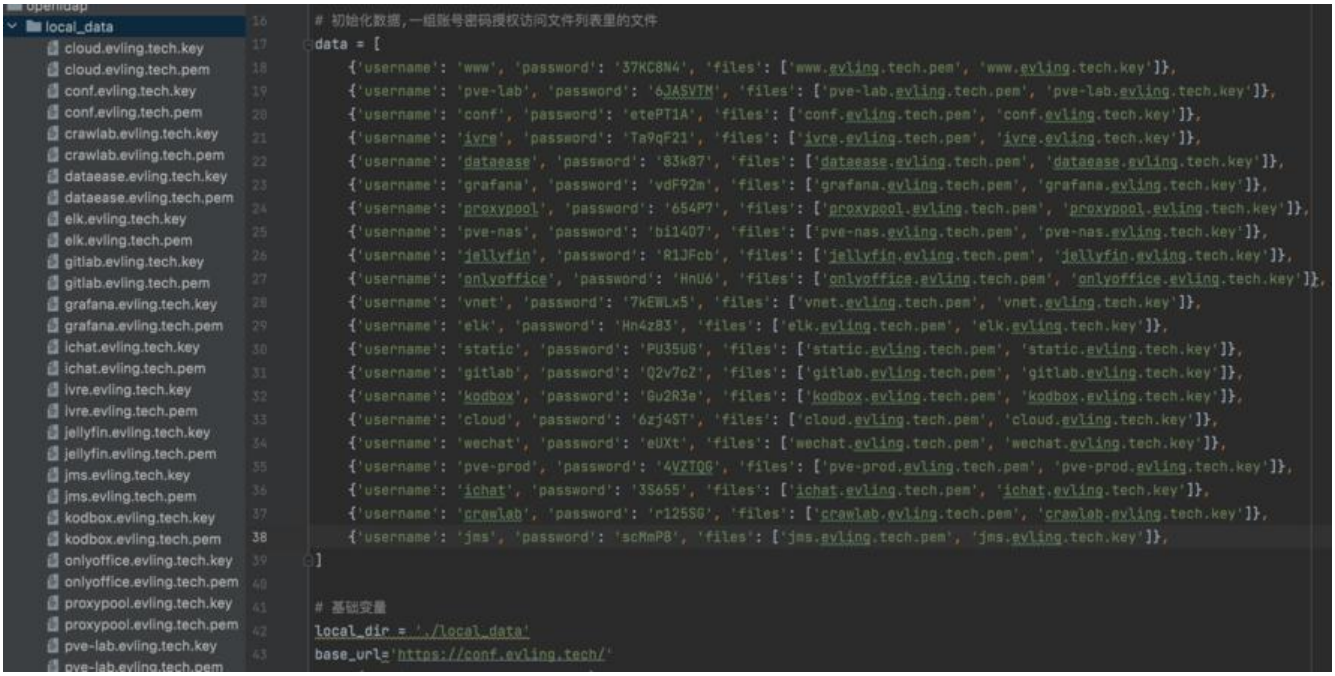
- /uploads/ path:filename

```
# 文件下载，仅普通用户有权限访问
auth=('admin', 'admin')    # 默认密码为admin
res = requests.get(base_url + "uploads/example.txt", auth=auth)
with open('example_local.txt', 'w') as f:
    f.write(res.content)
    f.close()
```

运维端维护示例脚本

假定服务端部署的地址为 https://conf.evling.tech，在客户端脚本所在目录创建一个目录./local_data 指定好相应的文件名，与脚本中的账号文件映射信息要匹配，如下图所示，主要是第一比较麻烦，后

更新就直接在本地当前目录结构进行变更即可，因为上传接口是通过覆盖的方式替换服务端文件的。




```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# @Project : tools
# @File : conf_mgr.py
# @Software: PyCharm
# @Author : 易雾君
# @Email : evling2020@gmail.com
# @公众号 : 易雾山庄
# @Site : https://www.evling.tech
# @Describe : 家庭基建，生活乐享.
# @Time : 2022/1/18 8:29 PM
import os.path

import requests

# 初始化数据,一组账号密码授权访问文件列表里的文件
data = [
    {'username': 'www', 'password': '37KC8N4', 'files': ['www.evling.tech.pem', 'www.evling.tech.
ey']},
    {'username': 'pve-lab', 'password': '6JASVTM', 'files': ['pve-lab.evling.tech.pem', 'pve-lab.evli
g.tech.key']},
    {'username': 'conf', 'password': 'etePT1A', 'files': ['conf.evling.tech.pem', 'conf.evling.tech.key
]},
    {'username': 'ivre', 'password': 'Ta9qF21', 'files': ['ivre.evling.tech.pem', 'ivre.evling.tech.key']}

    {'username': 'dataease', 'password': '83k87', 'files': ['dataease.evling.tech.pem', 'dataease.evl
ng.tech.key']},
    {'username': 'grafana', 'password': 'vdF92m', 'files': ['grafana.evling.tech.pem', 'grafana.evlin
.tech.key']},
    {'username': 'proxypool', 'password': '654P7', 'files': ['proxypool.evling.tech.pem', 'proxypool
evling.tech.key']},
    {'username': 'pve-nas', 'password': 'bi14D7', 'files': ['pve-nas.evling.tech.pem', 'pve-nas.evlin
.tech.key']},
```

```
    {'username': 'jellyfin', 'password': 'R1JFcb', 'files': ['jellyfin.evling.tech.pem', 'jellyfin.evling.te
h.key']},
    {'username': 'onlyoffice', 'password': 'HnU6', 'files': ['onlyoffice.evling.tech.pem', 'onlyoffice.
vling.tech.key']},
    {'username': 'vnet', 'password': '7kEWLx5', 'files': ['vnet.evling.tech.pem', 'vnet.evling.tech.ke
']},
    {'username': 'elk', 'password': 'Hn4z83', 'files': ['elk.evling.tech.pem', 'elk.evling.tech.key']},
    {'username': 'static', 'password': 'PU35UG', 'files': ['static.evling.tech.pem', 'static.evling.tech.
ey']},
    {'username': 'gitlab', 'password': 'Q2v7cZ', 'files': ['gitlab.evling.tech.pem', 'gitlab.evling.tech
key']},
    {'username': 'kodbox', 'password': 'Gu2R3e', 'files': ['kodbox.evling.tech.pem', 'kodbox.evling
tech.key']},
    {'username': 'cloud', 'password': '6zj4ST', 'files': ['cloud.evling.tech.pem', 'cloud.evling.tech.k
y']},
    {'username': 'wechat', 'password': 'eUXt', 'files': ['wechat.evling.tech.pem', 'wechat.evling.tec
.key']},
    {'username': 'pve-prod', 'password': '4VZTQG', 'files': ['pve-prod.evling.tech.pem', 'pve-prod
evling.tech.key']},
    {'username': 'ichat', 'password': '3S655', 'files': ['ichat.evling.tech.pem', 'ichat.evling.tech.key'
},
    {'username': 'crawlab', 'password': 'r125SG', 'files': ['crawlab.evling.tech.pem', 'crawlab.evling
tech.key']},
    {'username': 'jms', 'password': 'scMmP8', 'files': ['jms.evling.tech.pem', 'jms.evling.tech.key']},
]

# 基础变量
local_dir = './local_data'
base_url='https://conf.evling.tech/'
auth=('admin', 'your_admin_password')

for each_item in data:
    # 上传文件
    for file in each_item['files']:
        res = requests.post(base_url, files={'file': open(os.path.join(local_dir, file), 'rb')} ,auth=auth

        print(res.content)
    # 新建用户
    res = requests.post(url=base_url + "users/add", data={'username': each_item['username'], '
assword': each_item['password']}, auth=auth)
    print(res.status_code)
    print(res.json())
    # 授权文件访问
    res = requests.post(url=base_url + "users/authorize", json={'username': each_item['userna
e'], 'files': each_item['files']}, auth=auth)
    print(res.status_code)
    print(res.json())
```

消费客户端

消费客户端直接拿到上边给定的账号进行无脑下载即可，实现思路是，crontab定时任务，将服务端指定文件下载到本地的一个临时文件路径，用diff比对原文件与临时文件，不同即进行文件权限拷贝并替换。wget命令中记得密码改成你自己的，访问密码不建议以bash传参的形式，示例脚本：

```
#!/bin/bash
# ScriptName: conf_client.sh

echo "usage: -$0 file1 file2 url"
file1=$1
file2=$2
url=$3

wget --user=www --password=pzgE8e $url --no-check-certificate -O $file2

if [ -f $file1 ] && [ -f $file2 ]; then
  diff $file1 $file2 > /dev/null
  if [ $? != 0 ]; then
    echo "Different!"
      chmod --reference=$file1 $file2
    mv $file2 $file1
  else
    echo "Same!"
  fi
else
  echo "$file1 or $file2 does not exist, please check filename."
fi
```

crontab 配置任务

```
# 每天中午12点同步一下
0 12 * * *      /bin/bash/bash /data/tools/conf_client.sh /etc/letsencrypt/certs/www.evling.tech pem /tmp/www.evling.tech.pem https://conf.evling.tech/uploads/www.evling.tech.pem
```

下期

- 基于openldap实现账号统一管理