# SSO保护解决方案

## 背景

假如外围系统集成SSO没有修改好正确的错误跳转，那么出现错误的时候会极其频繁的调用SSO，然后因为某些机制的原因导致SSO宕机。

所以想通过监控Nginx的日志来看某段时间内访问SSO的次数是否正常来对SSO进行保护。

## 使用说明

### 文件简介

#### sso-protect.py

python主程序。通过读取配置文件获取到相关文件（如Nginx的访问日志，403页面等等）。监控Nginx访问日志的更新并判断某IP某段时间内的访问次数是否达到上限，如果达到上限便进行限制，同时更新403页面。

#### sso-protect-config.conf

配置文件。各配置说明如下：

- limit_time & limit_count

  配置单位时间和此事件内访问上限。默认5 & 100即5秒内某个IP能访问的上限是100次

- limit_service

  配置字符串标识。通过此标识确定IP是否访问的配置服务。如配置为"/ssoserver/login"，通过这个来匹配Nginx访问日志，来确定是否访问的SSO服务。

- limit_source

  配置Nginx访问日志的路径。

- location_deny

  Nginx黑名单的位置，主程序会把禁止的IP添加到此文件中。

- location_report

  展示页面，为了展示已经禁止访问SSO服务的IP。

#### 403.html

用于替换Nginx的403.html，当某IP禁止后跳转到此页面，看是不是因为访问达到上限。

## 开始使用

### 步骤一 Nginx新建黑名单并加入到SSO服务

如我们新建`deny.conf`并把它放在了Nginx的`conf`目录下后，添加`include`配置的语句，参照下面

```
location /ssoserver {
    include deny.conf;
    ... ...
}
```

### 步骤二 设置Nginx跳转403新页面

当错误码为403的时候，跳转到我们的禁止IP展示页面。配置如下

```
error_page  403  /403.html;
location = /403.html {
    root   E:\PythonProject\python-start\sso-protect;
}
```

配置完后再遇到403就会跳转到配置`root`下的403.html

### 步骤三 修改配置启动程序

修改`sso-protect-config.conf`文件然后启动`sso-protect.py`,这两个文件要放在同一级目录。

```
nohup python sso-protect.py &
```

## TODO

- 监控黑名单实现自动释放