# 手把手教你如何建立一个支持ctf动态独立靶机的靶场（ctfd+ctfd-whale）

原创

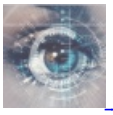
fjh1997 于 2019-09-15 14:58:21 发布 16718 收藏 58

分类专栏： 安全 运维 文章标签： ctf 动态靶机



本文链接：https://blog.csdn.net/fjh1997/article/details/100850756





安全 同时被 2 个专栏收录

54 篇文章 0 订阅

订阅专栏

运维

64 篇文章 1 订阅

订阅专栏

## 前言

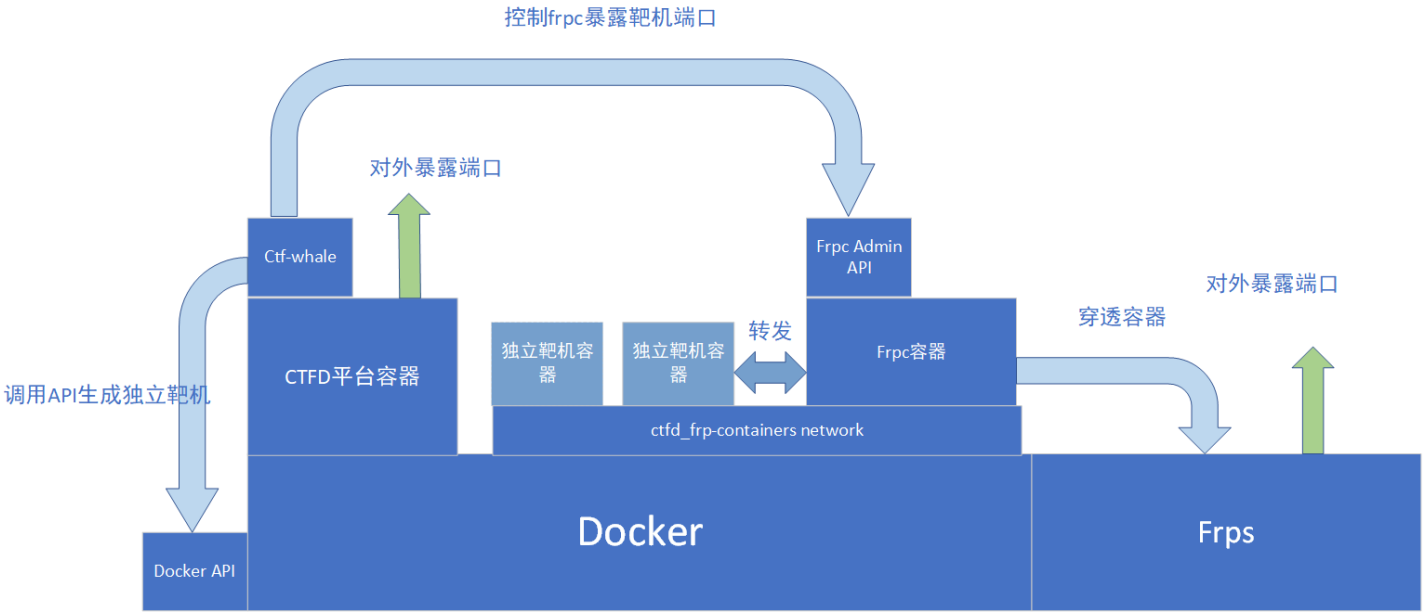
要说开源的ctf训练平台，ctfd是不错的选择，支持各种插件，今天我就来介绍一下如何部署赵今师傅为ctfd平台写的一款支持独立动态靶机的插件。前提是你的ctfd也是docker部署的。

## 动态独立靶机

说到独立动态靶机，各位ctfer可能会联想到i春秋举办的各类比赛，其中比赛环境大都用到了动态独立靶机（dynamic standalone instance）技术，也就是每做一道题，就会自动生成一个虚拟题目环境，每一个环境刚刚生成的时候都是崭新的，且能为每个队伍生成一个独一无二的flag，防止flag分享作弊的行为。

## 技术原理

先来看张图 = =




别被这张图吓到了， 这个图只是在本地部署动态靶机，而赵师傅的buuctf平台由于使用学校的服务器生成独立靶机还增加了内网穿透功能，网络结构要更复杂，所以这个插件有些功能是针对赵师傅下需求设计的，而我们仅仅只需要本地生成独立靶机即可，因此只需使用该插件如图所示的部分功能即可。

## 官方教程

建议先去看看官方教程：）本教程对应官方教程第一种模式。
https://www.zhaoj.in/read-6333.html

## 安装步骤

### 1.安装启动frps

下载frp并安装

```
cd
wget https://github.com/fatedier/frp/releases/download/v0.36.2/frp_0.36.2_linux_amd64.tar.gz
tar -zxvf frp_0.36.2_linux_amd64.tar.gz
cd frp_0.36.2_linux_amd64
sudo cp systemd/* /etc/systemd/system/
sudo mkdir /etc/frp
sudo cp frpc.ini  frps.ini /etc/frp/
sudo cp frpc  frps /usr/bin/
sudo chmod a+x /usr/bin/frpc /usr/bin/frps
sudo systemctl enable frps
```

编辑frps.ini

```
sudo vim /etc/frp/frps.ini
```

#### frps.ini

```
[common]
bind_port = 7897
bind_addr = 0.0.0.0
token =thisistoken
vhost_http_port=80 #如果是http动态域名需要这个。80端口开启需要systemmd使用root权限启用frp
```

至于frpc.ini，先不急着配置，后面会配置

启动frps系统服务

```
sudo systemctl start frps
```

## 2.创建网络并启动frpc容器并配置frpc.ini

启动frpc容器

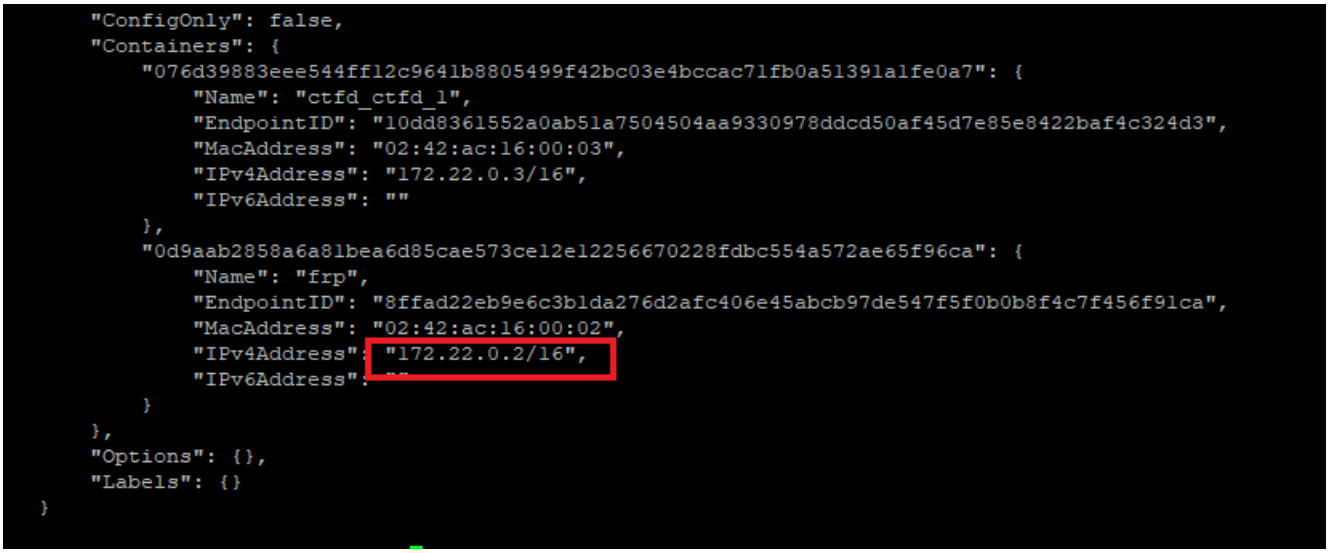
```
sudo docker network create ctfd_frp-containers
sudo docker run  -d -v ~/frp_0.36.2_linux_amd64/frpc.ini:/etc/frp/frpc.ini --network="ctfd_frp-containers" --res
tart=always "glzjin/frp"
```

创建网络frpcadmin用于ctfd容器和frpc容器通信

```
sudo docker network create frpcadmin
sudo docker network connect frpcadmin <frpc容器名或者ID>
#将frpc和ctfd容器单独连接到网络frpcadmin,注意要等容器创建好之后再连接，因此ctfd将在稍后连接
```

查看frpcadmin网络的连接情况并记录frpc容器的网络IP

```
docker network inspect frpcadmin
```

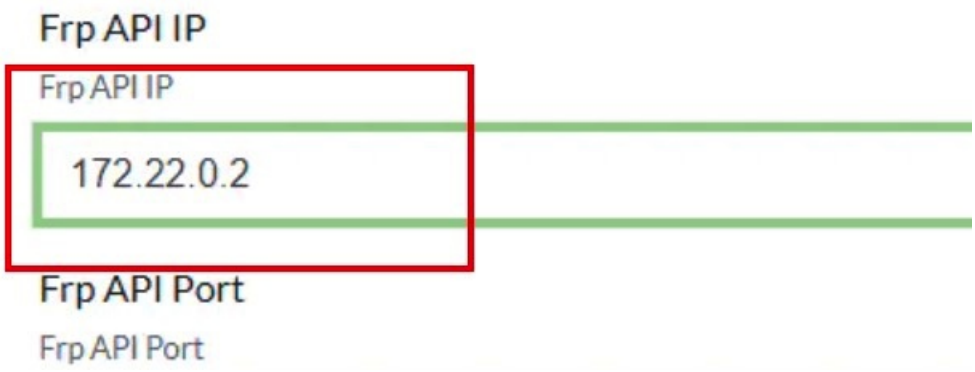



编辑frpc.ini

```
vim ~/frp_0.36.2_linux_amd64/frpc.ini
```

### frpc.ini

```
[common]
server_addr = 172.17.0.1 # 这里填写宿主机ifconfig之后docker0的ip，因人而异，不要一摸一样填
server_port = 7897
token=thisistoken
admin_addr = 172.22.0.2 #这里填写frpc容器在frpcadmin网络里的ip，因人而异，不要一摸一样填，而且要和后续下图插件配置界面中
的一样。
admin_port = 7400
log_file = ./frpc.log
```




编辑完之后记得重启frpc容器

```
docker restart <frpc容器的ID>
```

## 3.安装靶场与插件

下载靶场与插件

```
git clone https://github.com/CTFd/CTFd.git
cd CTFd/
git reset 6c5c63d667a17aec159c8e26ea53dccfbc4d0fa3 --hard
#回滚到当前教程适合的版本
cd CTFd/plugins #打开ctfd插件目录
git clone https://github.com/glzjin/CTFd-Whale.git ctfd-whale #确保插件文件夹小写
cd ctfd-whale
git reset 5b32f457e9f56ee9b2b29495f4b3b118be3c57bd --hard #回滚到当前教程适合的版本
cd ../.. #返回ctfd主目录
vim docker-compose.yml
```

配置docker-compose.yml

```yaml
version: '2.2'

services:
  ctfd:
    build: .
    user: root
    restart: always
    ports:
      - "8000:8000" #你自己的
    environment:
      - UPLOAD_FOLDER=/var/uploads
      - DATABASE_URL=mysql+pymysql://root:ctfd@db/ctfd
      - REDIS_URL=redis://cache:6379
      - WORKERS=1
      - LOG_FOLDER=/var/log/CTFd
      - ACCESS_LOG=-
      - ERROR_LOG=-
    volumes:
      - .data/CTFd/logs:/var/log/CTFd
      - .data/CTFd/uploads:/var/uploads
      - .:/opt/CTFd:ro
      - /var/run/docker.sock:/var/run/docker.sock #添加这句即可，别的基本按照官方的不用动
    depends_on:
      - db
    networks:
        default:
        internal:

  db:
    image: mariadb:10.4.12 #这里改成10.4.12，10.4.13会出错
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=ctfd
      - MYSQL_USER=ctfd
      - MYSQL_PASSWORD=ctfd
      - MYSQL_DATABASE=ctfd
    volumes:
      - .data/mysql:/var/lib/mysql
    networks:
        internal:
    # This command is required to set important mariadb defaults
    command: [mysqld, --character-set-server=utf8mb4, --collation-server=utf8mb4_unicode_ci, --wait_timeout=2880
0, --log-warnings=0]

  cache:
    image: redis:4
    restart: always
    volumes:
    - .data/redis:/data
    networks:
        internal:

networks:
    default:
        external:
          name: frpcadmin
    internal:
        internal: true
```

重新build后启动ctfd
如果是在国内网络环境下构建镜像的同学，建议修改Dockerfile为以下来使用豆瓣源or阿里源:

```
FROM python:2.7-alpine
RUN sed -i 's/dl-cdn.alpinelinux.org/mirrors.aliyun.com/g' /etc/apk/repositories &&\
    apk update && \
    apk add python python-dev linux-headers libffi-dev gcc make musl-dev py-pip mysql-client git openssl-dev g++
RUN adduser -D -u 1001 -s /bin/bash ctfd

WORKDIR /opt/CTFd
RUN mkdir -p /opt/CTFd /var/log/CTFd /var/uploads
RUN pip config set global.index-url https://pypi.doubanio.com/simple
RUN pip config set install.trusted-host pypi.doubanio.com
COPY requirements.txt .

RUN pip install -r requirements.txt -i  https://pypi.doubanio.com/simple

COPY . /opt/CTFd

RUN for d in CTFd/plugins/*; do \
      if [ -f "$d/requirements.txt" ]; then \
        pip install -r $d/requirements.txt -i  https://pypi.doubanio.com/simple; \
      fi; \
    done;

RUN chmod +x /opt/CTFd/docker-entrypoint.sh
RUN chown -R 1001:1001 /opt/CTFd
RUN chown -R 1001:1001 /var/log/CTFd /var/uploads

USER 1001
EXPOSE 8000
ENTRYPOINT ["/opt/CTFd/docker-entrypoint.sh"]
```

然后部署下阿里云的docker加速器：
https://help.aliyun.com/document_detail/60750.html

https://cr.console.aliyun.com/cn-qingdao/instances/mirrors
选择左下角镜像加速器

注意由于这篇文章年代久远banal的版本已经不支持python2。如果遇到以下问题：

> def is_sequence(obj: Any) -> bool syntax error:

请在reqiurements.txt添加：

```
banal==0.4.2
```

如果遇到以下问题：

> ERROR: Package ‘pyrsistent’ requires a different Python: 2.7.18 not in ‘>=3.5’

请在reqiurements.txt添加：

```
pyrsistent==0.14.0
```

如果遇到问题：

> ERROR: botocore 1.12.253 has requirement urllib3<1.26,>=1.20; python_version == “2.7”, but you’ll have urllib3 1.26.4 which is incompatible.
> 请在reqiurements.txt添加：

```
urllib3==1.25.11
```

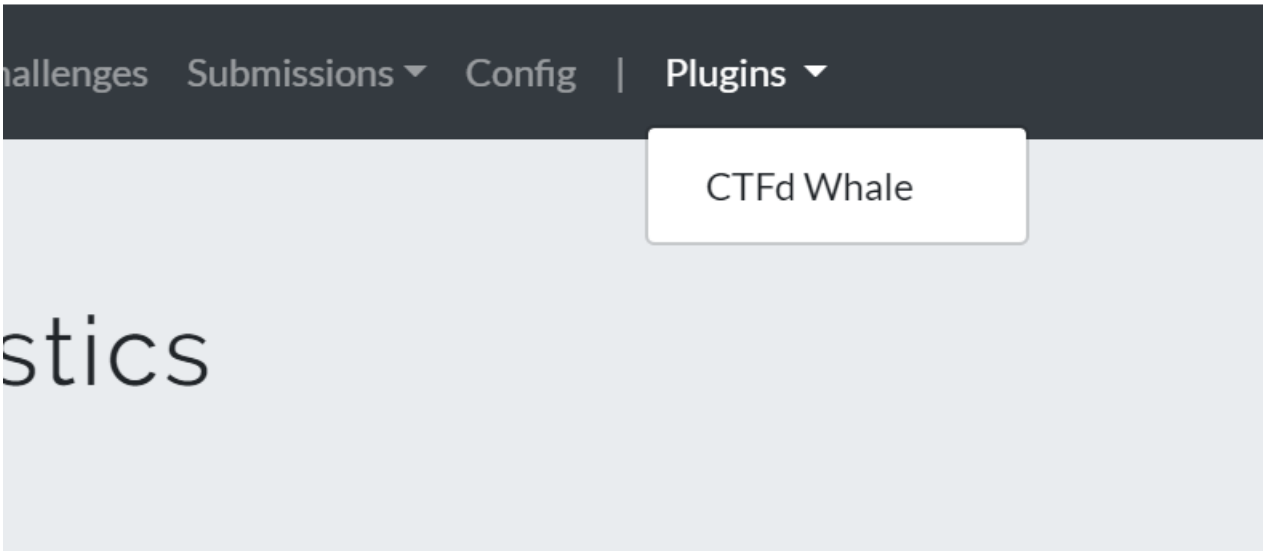
之后构建启动镜像。

```
docker-compose build
docker-compose up -d
```

将ctfd连接frpcadmin网络（如果使用前面的docker-compose.xml，这步可以忽略）

```
docker network connect frpcadmin <ctfd容器名或者ID>
```

启动ctfd后进入管理页面选择插件




填写相关参数，注意要在template里面写上frpc.ini的内容。里面domain填写自己的

Docker API URL

Docker API URL to connect

unix:///var/run/docker.sock

Frp API IP

Frp API IP

172.22.0.2

Frp API Port

Frp API Port

7400

Frp Http Domain Suffix

Docker API URL to connect

.ctf.fjh1997.top

Frp Direct IP Address

For direct redirect

52.82.46.169

Frp Direct Minimum Port

For direct redirect

28000

Frp Direct Maximum Port

For direct redirect

29000

Max Container Count

Max Container Count

30

Max Renewal Times

Max Renewal Times

20

Frp config template

Frp config template, only need common section!

```
[common]
server_addr = 172.17.0.1
server_port = 7897
token=fjh19971030
admin_addr = 172.22.0.2
admin_port = 7400
log_file = ./frps.log
```

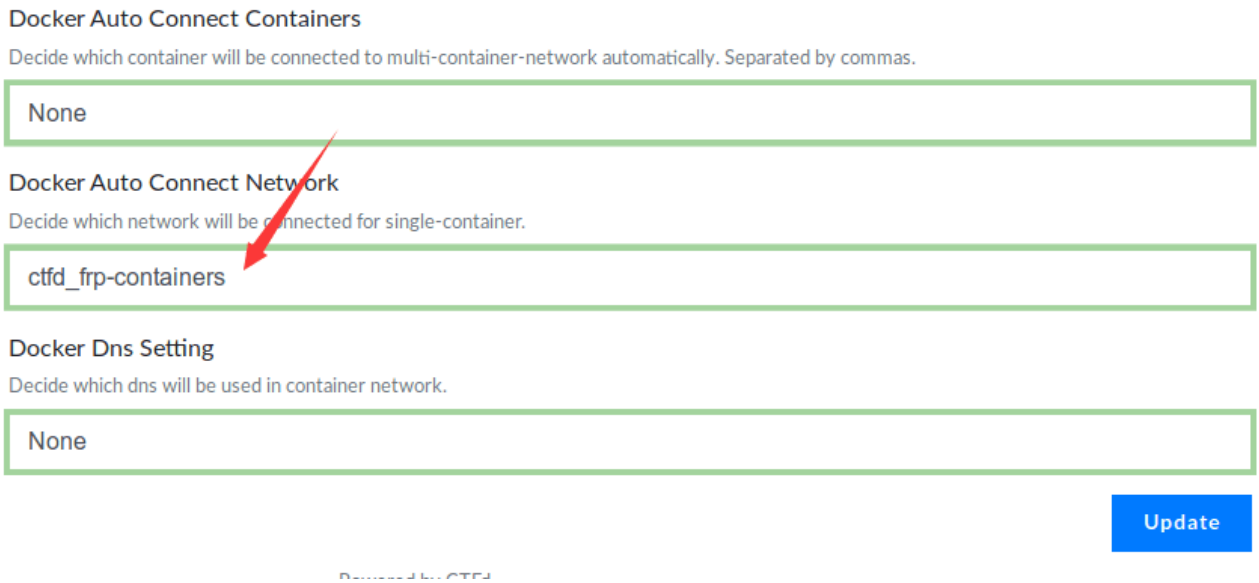
Docker Auto Connect Containers

Decide which container will be connected to multi-container-network automatically. Separated by commas.

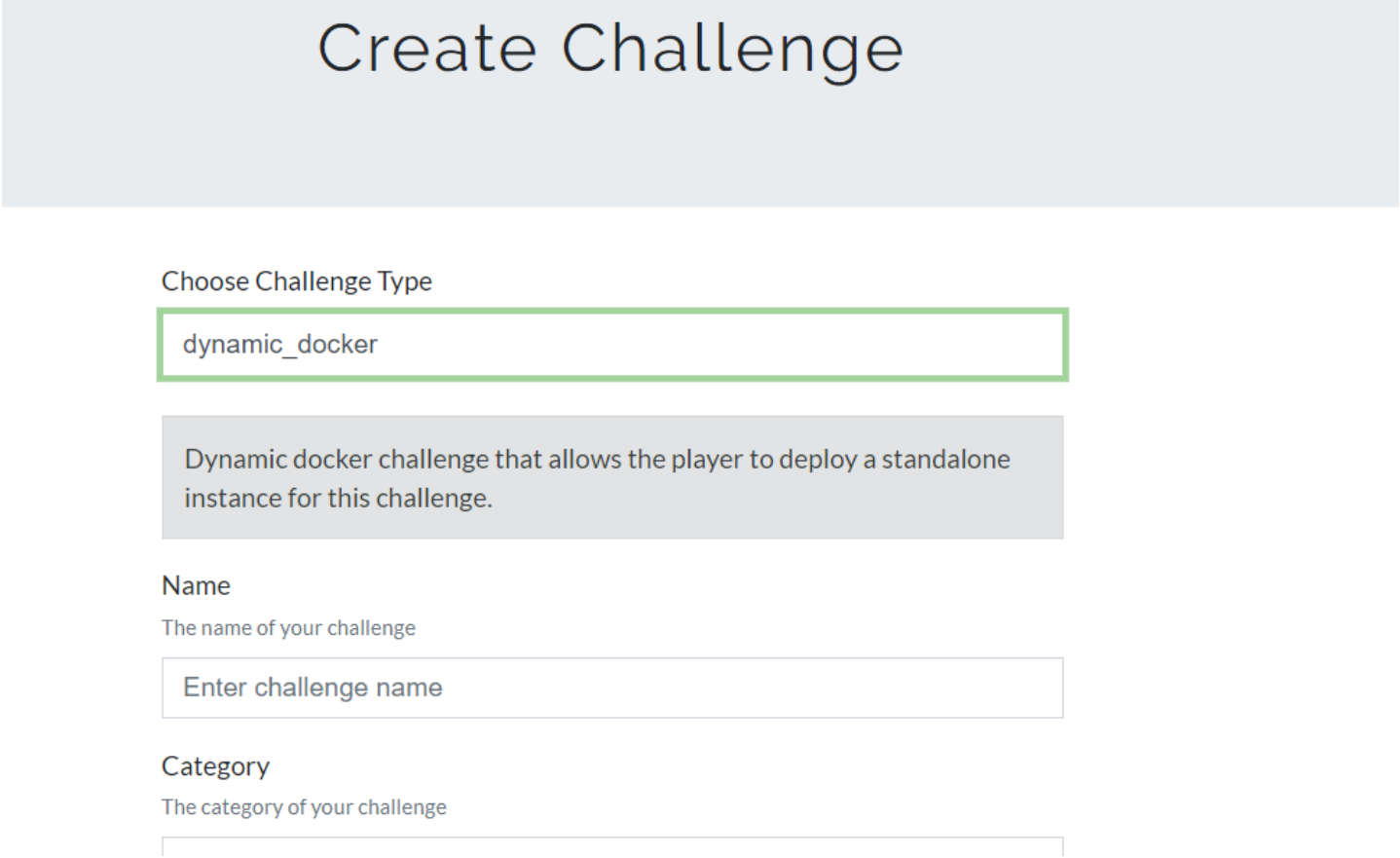
None

Update

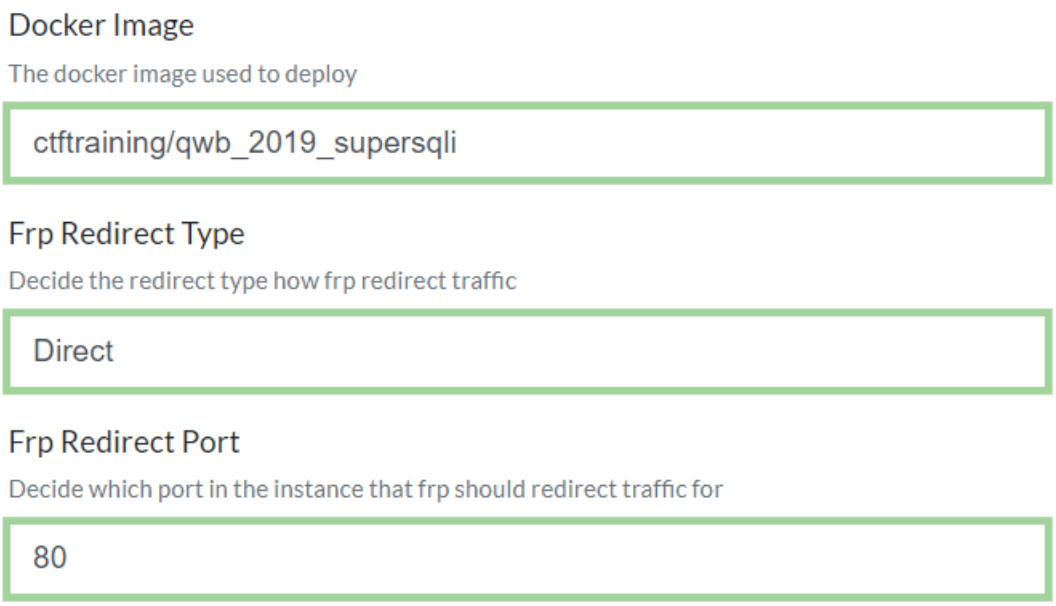
由于我这篇文章写的比较早，ctfd-whale已经有了很多次更新，根据gtfly师傅的提醒，这里需要填写创建的网络。




启动题目选择dynamic_docker




这里选择一个支持动态FLAG的docker镜像，并选择设置该docker镜像的内部端口




内部端口可以用

docker ps查看

```
f3e00a9f23d8        ctftraining/qwb_2019_supersqli   "docker-php-entrypoi…"
   8 seconds ago       Up 5 seconds        80/tcp, 9000/tcp
 1-09077a27-3590-4cd9-b1cc-8c7483190c09
```

## 3b.使用docker-compose一站式部署靶场、容器与网络

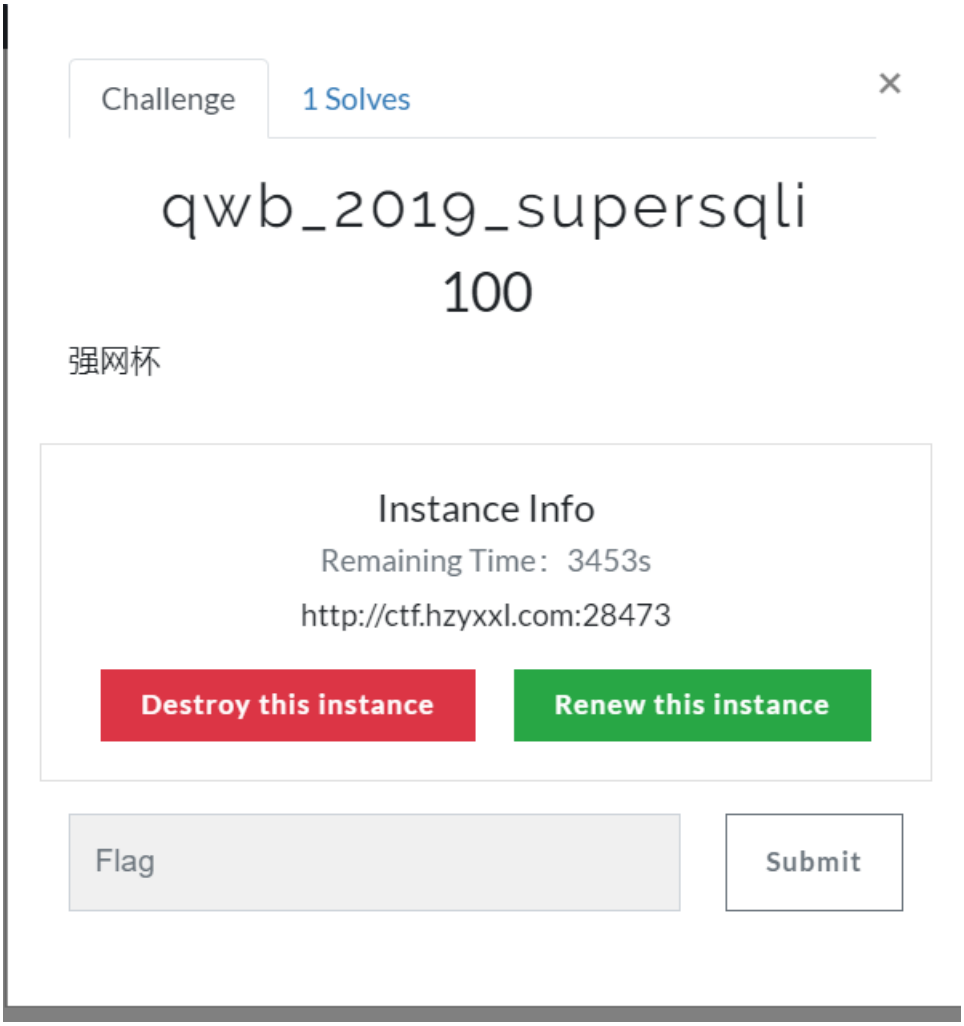
以下docker-compose.yml的功能与上述操作步骤一样的，作为可选项，有能力的同学可以试试看

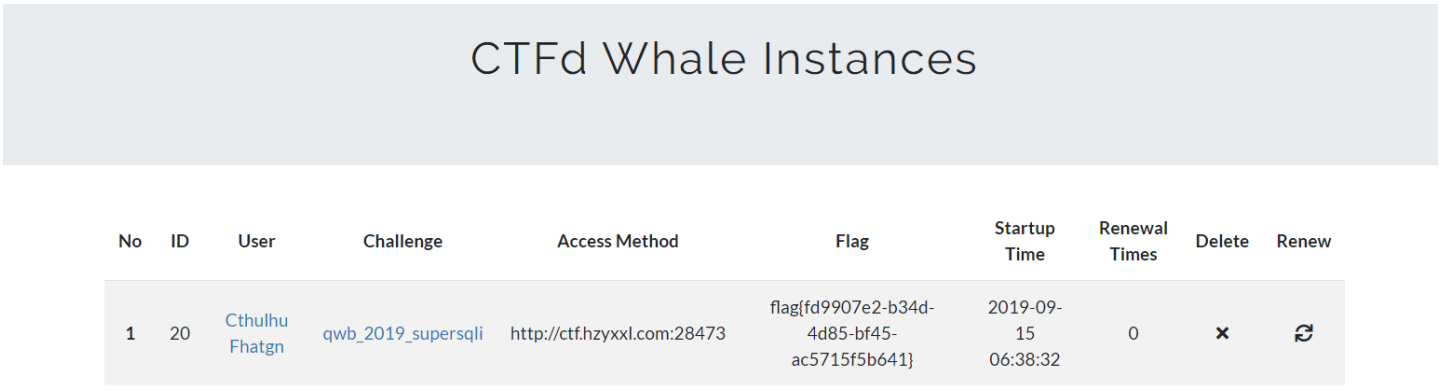
```yaml
version: '2'

services:
  ctfd:
    build: .
    user: root
    restart: always
    ports:
      - "8000:8000" #你自己的
    environment:
      - UPLOAD_FOLDER=/var/uploads
      - DATABASE_URL=mysql+pymysql://root:ctfd@db/ctfd
      - REDIS_URL=redis://cache:6379
      - WORKERS=1
      - LOG_FOLDER=/var/log/CTFd
      - ACCESS_LOG=-
      - ERROR_LOG=-
    volumes:
      - .data/CTFd/logs:/var/log/CTFd
      - .data/CTFd/uploads:/var/uploads
      - .:/opt/CTFd:ro
      - /var/run/docker.sock:/var/run/docker.sock #添加这句即可，别的基本按照官方的不用动
    depends_on:
      - db
    networks:
        default:
        internal:
```

```
        internal:
        frpcadmin:
            ipv4_address: 172.22.0.3

  db:
    image: mariadb:10.4.12
    restart: always
    environment:
      - MYSQL_ROOT_PASSWORD=ctfd
      - MYSQL_USER=ctfd
      - MYSQL_PASSWORD=ctfd
      - MYSQL_DATABASE=ctfd
    volumes:
      - .data/mysql:/var/lib/mysql
    networks:
        internal:

    # This command is required to set important mariadb defaults
    command: [mysqld, --character-set-server=utf8mb4, --collation-server=utf8mb4_unicode_ci, --wait_timeout=2880
0, --log-warnings=0]

  cache:
    image: redis:4
    restart: always
    volumes:
    - .data/redis:/data
    networks:
        internal:
  frpc:
    image: glzjin/frp:latest
    restart: always
    volumes:
      - ~/frp_0.36.2_linux_amd64/frpc.ini:/etc/frp/frpc.ini
    networks:
        frpcadmin:
            ipv4_address: 172.22.0.2
        frp-containers:
networks:
    default:
    internal:
        internal: true
    frpcadmin:
        driver: bridge
        internal: true
        ipam:
            config:
                - subnet: 172.22.0.0/16
    frp-containers:
        driver: bridge
        ipam:
            config:
                - subnet: 172.21.0.0/16
```

## 4.测试

不出意外的话，现在在challenge界面可以使用动态靶机
管理员也可以在后台管理靶机




## 5.制作支持动态flag的镜像

请看gtfly写的

http://www.gtfly.top/2019/09/27/CTFd%E5%8A%A8%E6%80%81docker%E9%95%9C%E5%83%8F%E7%BC%96%E5%86%99.html

或者可以看看赵师傅的贡献指南。
https://www.zhaoj.in/read-6259.html

推荐一个github靶机仓库

https://github.com/CTFTraining

## pwn题

1、创建一个新目录。

```
mkdir pwntest
cd pwntest
```

2、进入该目录，将可执行文件拷贝到该目录下，命名为 pwn。创建一个 Dockerfile 文件，内容如下：

```
FROM glzjin/pwn_base_18
COPY pwn /pwn/pwn
```

glzjin/pwn_base_18 代表 Ubuntu 18.04, glzjin/pwn_base_16 代表 Ubuntu 16.04, glzjin/pwn_base_19 代表 Ubuntu 19.04, glzjin/pwn_base_20 代表 Ubuntu 20.04。注意，pwn的端口默认是10000

3、构建。

```
docker build -t <你在 dockerhub的用户名/你的镜像名> ./
```

## 常见错误分析与检查

### 一般错误：

可以使用命令`docker logs <ctfd容器的ID>`查看报错情况

### docker容器无法启动：

确保docker api填写正确，如示例中为unix:///var/run/docker.sock
你也可以使用端口形式的api如官方示例：可以用IP：端口指定API
然后使用如下命令进入ctfd容器，手动调用端口测试

```
docker exec -it <ctfd容器的ID> /bin/sh
/opt/CTFd# python
```

```
>>>import docker
>>>client=docker.DockerClient(base_url="unix:///var/run/docker.sock")
>>>client.images.list()
```

如果api正确会列出所有镜像

### frp端口无法映射：

可以使用如下命令进入ctfd容器，手动调用端口测试

```
docker exec -it <ctfd容器的ID> /bin/sh
/opt/CTFd# python
```

```
>>>import requests
>>>requests.get("http://172.22.0.2:7400/api/reload")
<Response [200]> #这个表示成功
```

如果还是不行，也可以尝试查看ctfd和frpc容器的日志来分析解决。

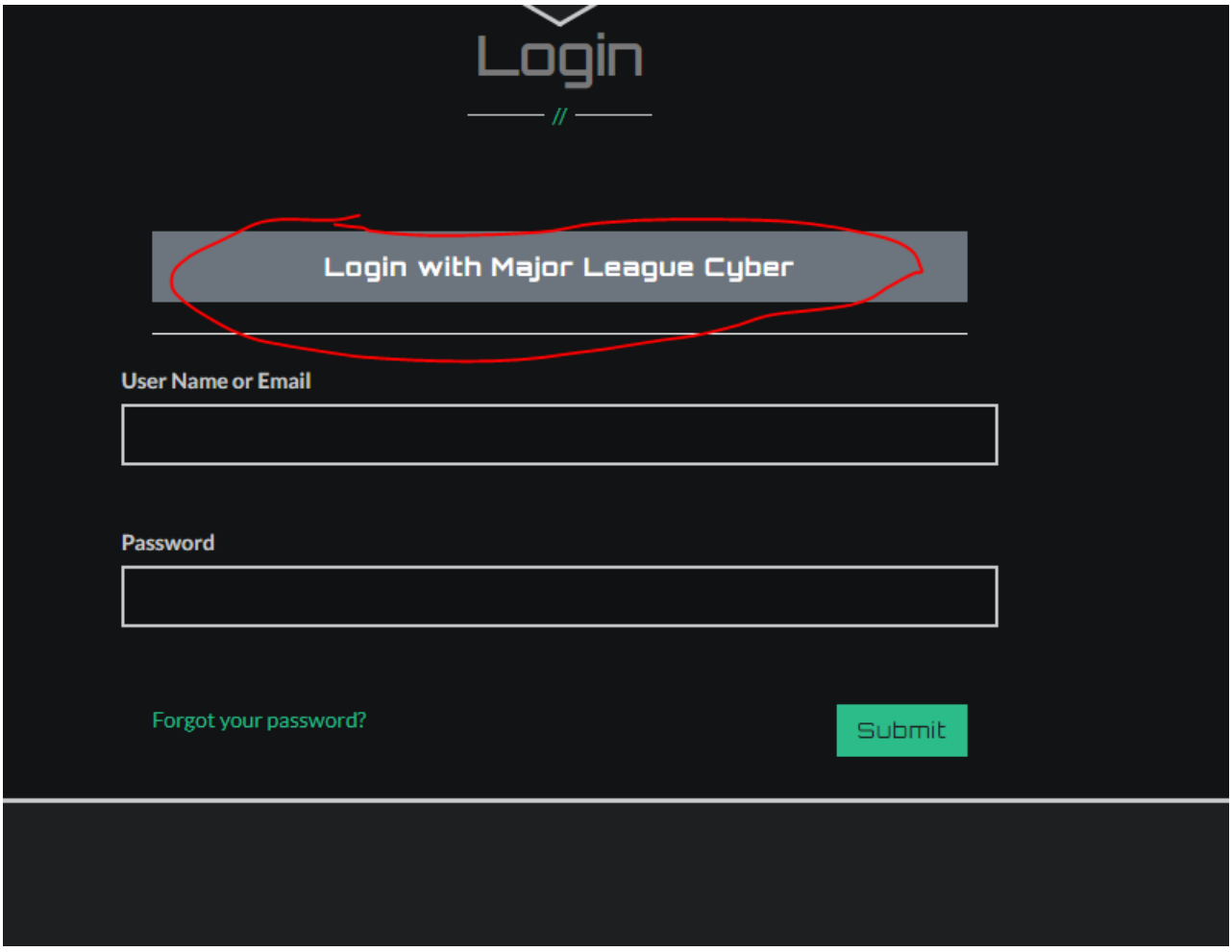
```
docker logs <ctfd容器的ID>
docker logs <frpc容器的ID>
```

```
docker exec -it <frpc容器的ID> /bin/sh
/etc/frpc# cat frpc.log
/etc/frpc# cat /tmp/frpc.ini
```

或者在ctfd源码里面使用print打印日志，再重启容器即可。

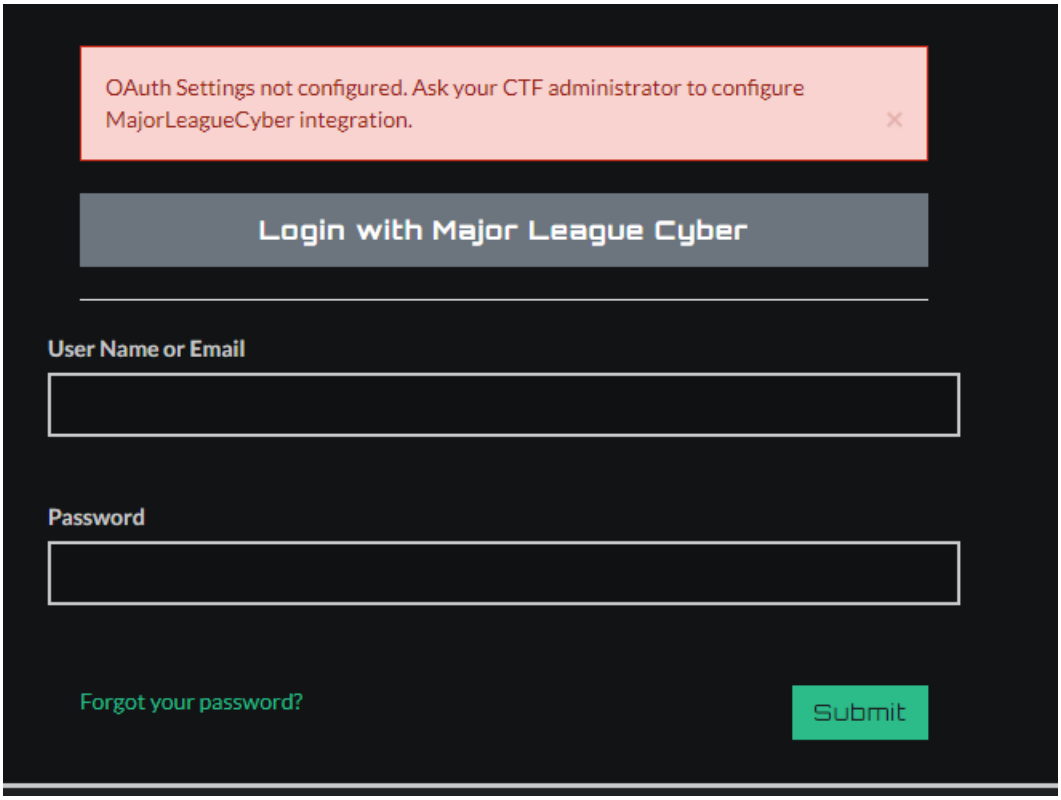
## 题外话

在搭建ctfd的时候，默认会有一个类似微信登录的方式登录的选项，就是使用MLC来登录。

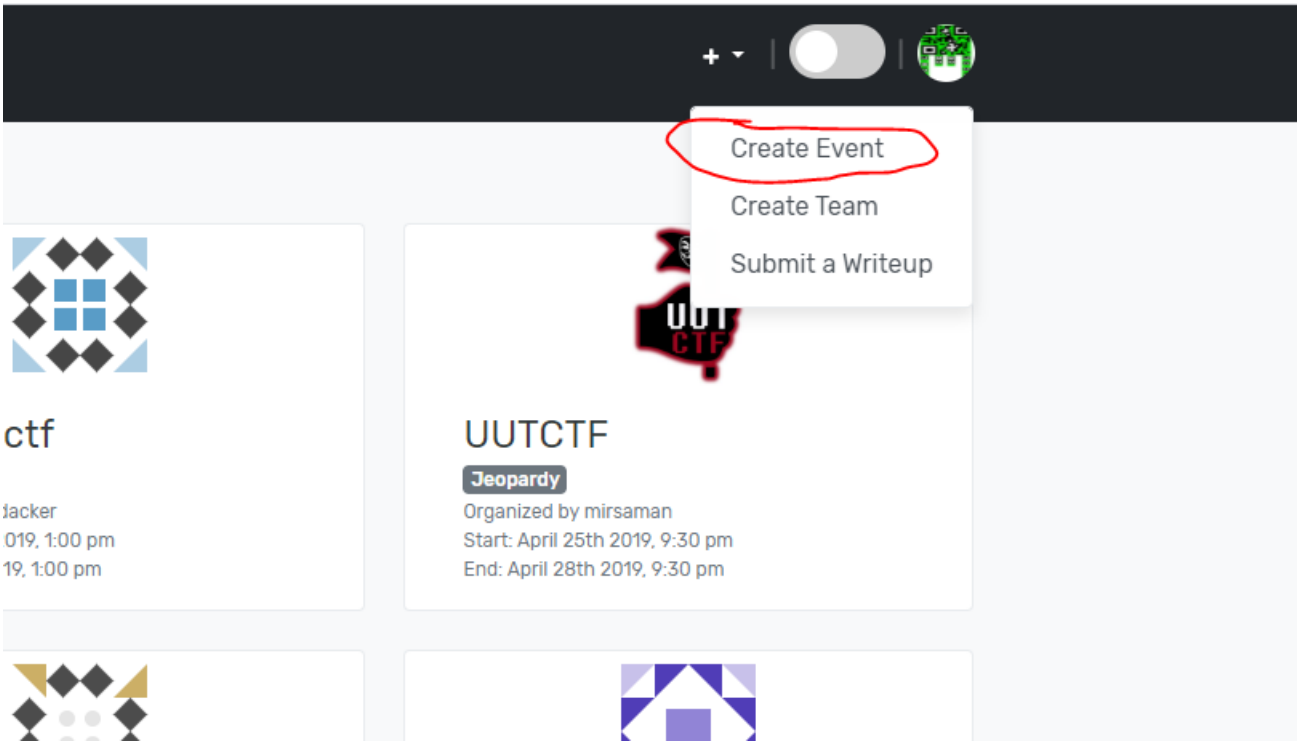



但是当你试图用mlc登录的时候，你就会收到这样的错误

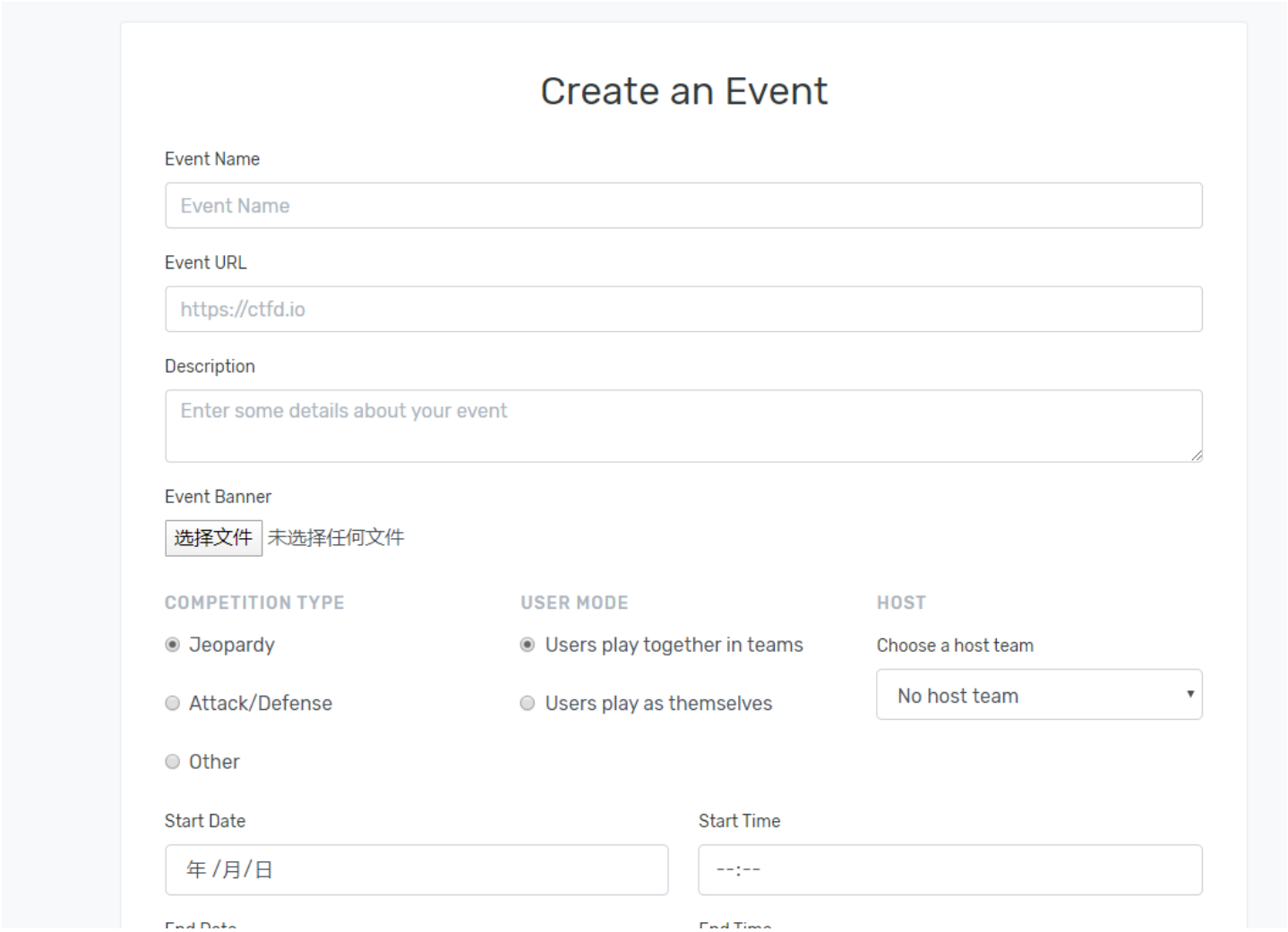
> OAuth Settings not configured. Ask your CTF administrator to configure MajorLeagueCyber integration.



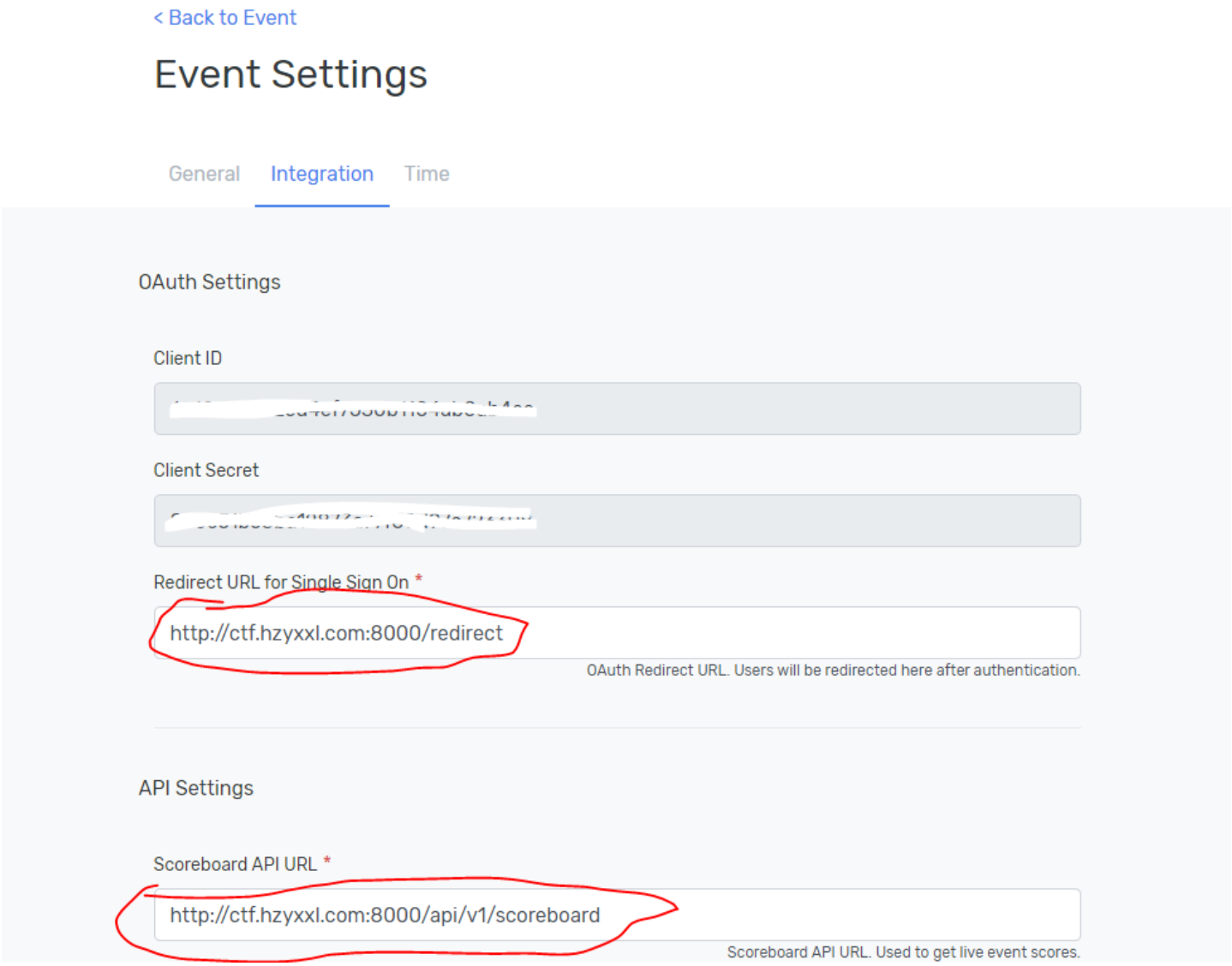

下面教你怎么开启使用mlc登录的功能
首先注册并登录mlc



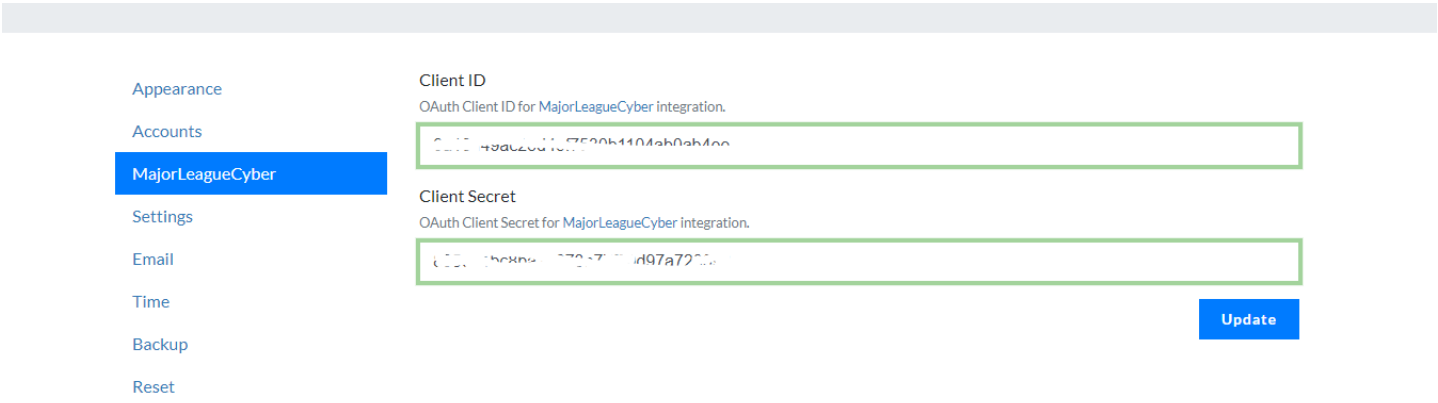

选择create event




创建好了以后按照下面的方式填写相关信息




并把上面的Client ID和secret填写到ctfd的设置页面




之后就可以用mlc来登录了，当然登录的前提是用户已经注册mlc的账号，使用mlc账号登录的话会有一个official的标签如下：

| ID | User | Email | Website | Country | Admin | Verified | Hidden | Banned |
|---|---|---|---|---|---|---|---|---|
| 52 | YSC621 | 1849297531@qq.com | | | | | | |
| 53 | fjh1997 Official | fjhhz1997@gmail.com | | | | verified | | |
| 54 | peri0d | peri0d1@foxmail.com | | | | | | |
| 55 | 614751780 | 614751780@qq.com | | | | | | |
| 56 | 洛颖辰 | 761791827@qq.com | | | | | | |
| 57 | mile | 2649190654@qq.com | | | | | | |

是不是很帅呢？

## 友情连接

BUUCTF

Cthulhu OJ

## 获取帮助

如果配置有遇到困难可以加入交流群:729021148
或者本人淘宝店：https://item.taobao.com/item.htm?id=612013759404