# AWD 训练平台

## 简介

平台由三个核心组件构成，Dockerfile生成的若干容器作为gamebox，Flask HttpServer来收集参赛者提交的flag并展示分数，Redis数据库负责存储各个gamebox上的flag，统一干里所有flag和team的对应关系。

## 结构

init.py用于在比赛最开始时初始化，根据输入的比赛队伍数量生成对应数量的初始化文件

awd.log记录比赛事件，回合数，得分情况等等

controller.py是Flask app的核心脚本，内部处理所有的http请求。

genflag.sh负责定期生成各个队伍的flag。

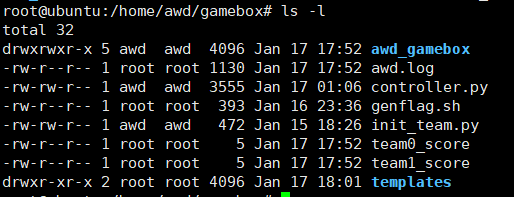
Awd\_gamebox文件夹中

Dockerfile是gamebox的dockerfile

project中bin文件夹中存放pwn题，html文件夹中存web题

ctf.xinetd为pwn题的服务配置文件

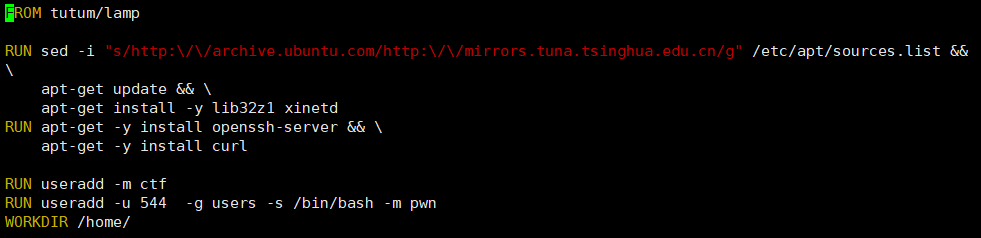
ssh.config是ssh配置文件



## Dockerfile

Gamebox的主体是一个精简的LAMP ubuntu docker。

改造工作主要是将awd必须的程序文件拷贝进去并提前安装一些必须的程序，比如curl，openssh，xinetd。



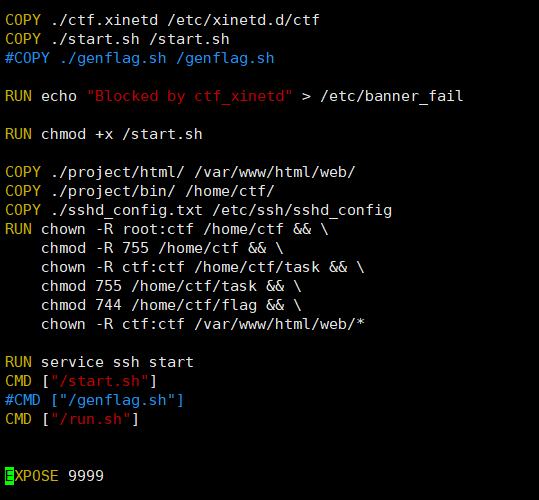
主要是用四个用户，root为维护账号、ctf为选手防守是用账号、pwn为攻击选手可以获得的权限账户，www-user为web攻击者可以获得的权限。

pwn题目在home/ctf文件夹中，web题目在/var/www/html文件夹中。

flag在同路径下flag文件夹中。

ctf用户可以任意修改pwn文件和web文件，pwn用户仅拥有pwn题执行权限，www-user同样可以修改web文件。

Docker启动时会同时启动web和pwn题的服务，运行docker时需要指定绑定的端口。目前还没有实现在模拟同网断内的机器互联。



## controller.py



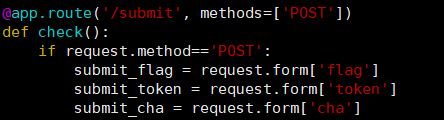
默认5000端口目前提供3个api

接收更新flag的api

本地更新flag之后会通过这个文件来讲flag更新到redis数据库当中，保持数据库中的flag为最新

提交flag的api：submit

队伍拿到flag之后通过这个接口提交，服务端会判断flag的正确性并更新得分情况



需要参数为本队token、拿到的flag、对应的题目

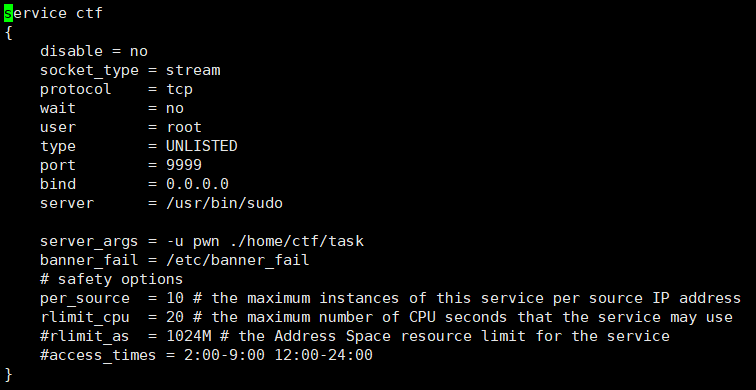
获取得分的api：socre

队伍可以随时通过这个接口查询所有队伍的得分



## ctf.xinetd

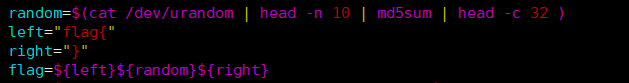
pwn题的服务文件，讲题目作为服务绑定到对应端口，以pwn用户执行。



## genflag.sh

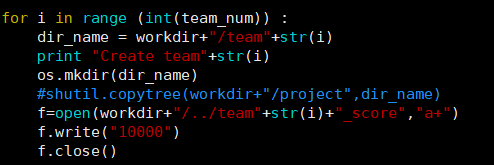
是用随机数生成flag，存储到对应team的文件夹，并向服务端发送请求更新flag

ctf和pwn用户都没有权限删除flag



## init.sh

初始化每个队伍的得分文件，初始化每个队伍的flag文件夹。



## 使用

清空redis

docker build -t "gamebox" .

init.py确定队伍数量

genflag.sh修改队伍数量

新建docker

docker run -d -v /home/awd/gamebox/awd\_gamebox/team0:/home/ctf/flag -p 18080:80 -p 13306:3306 -p 10022:22 -p 19999:9999 -h "gamebox0" --name="gamebox0" gamebox

docker run -d -v /home/awd/gamebox/awd\_gamebox/team1:/home/ctf/flag -p 18081:80 -p 13307:3306 -p 10023:22 -p 20000:9999 -h "gamebox1" --name="gamebox1" gamebox

开启服务

修改密码

启动flask

运行genflag.sh

开始比赛

## 效果

提交错误flag



提交自己的flag



提交正确



重复提交



查看分数

