CTF Web出题感悟

原创

Wuuconix 于 2021-03-31 20:34:32 发布 1270 收藏 27

文章标签： flag web docker 靶机

Wuuconix wanna a girlfriend!

本文链接：https://blog.csdn.net/Cypher_X/article/details/115359957

CTF Web出题感悟

最近帮忙学校ctf新生赛出题，负责出了web的8道题，8道题都很水，但是在出题时学习到了很多Docker镜像方面的知识。

首先我们来看看一道题是怎么组成的。

比如这道F12看源码的题目。

```
wuuconix@wuuconix ~ tree -a F12
F12
├── docker-compose.yml
├── Dockerfile
└── files
    ├── flag.sh
    └── html
        └── index.php
```

在实际过程中，个人感觉docker-compose.yml可有可无，所以我们主要来介绍以下Dockerfile flag.sh 和 index.php

很显然，index.php就是做题时的主界面。

```
<p style="font-family:arial;color:black;font-size:20px;text-align:center;">Flag不在这里哦</p>
<!-- flag{y0u_4re_mas7er_1n_v1ew_sourc4} -->
</body>
<script>
 document.oncontextmenu=function() {
   alert("右键被禁用");
   return false;
  };
 document.onkeydown = function(e) {
    e = window.event || e;
    var k = e.keyCode;
    //屏蔽ctrl+u，F12键
    if ((e.ctrlKey == true && k == 85) || k == 123) {
      if (k == 85)
        alert("Ctrl+U被禁用!");
      else
        alert("F12被禁用!");
      e.keyCode = 0;
      e.returnValue = false;
      e.cancelBubble = true;
      return false;
    }
 }
</script>
```

```
</script>
```

功能和考点显而易见，不予解释。我们再来看看Dockerfile

```
FROM ctftraining/base_image_nginx_mysql_php_73

LABEL Author="wuuconix <wuuconix@gmail.com>"
LABEL Blog="https://blog.csdn.net/Cypher_X"

COPY ./files /tmp/
RUN cp -rf /tmp/html /var/www/ \
    && cp -f /tmp/flag.sh /flag.sh \
    && chown -R www-data:www-data /var/www/html \
```

FROM表示基础镜像是从哪里来的 我们这里使用了 ctftraining/base_image_nginx_mysql_php_73 这个镜像，它包含nginx，mysql和7.3版本的php，而是不用自己写nginx.conf配置文件之类的，十分方便。

LABEL相当于给这个Dockerfile打上一个标签，标明作者名字，作者邮箱，和博客。

COPY就是把当前文件目录里的files文件夹复制到容器内部的/tmp文件夹里。这里要注意这个动作是发生在宿主机和docker容器之间的，要区别去接下来RUN命令里的cp，那里面的cp发生在docker容器内部。

之后的RUN相当在容器内部来执行命令，我们看到把html文件夹以-rf的方式（迭代且覆盖）的方式复制到www文件夹中，并且把flag.sh放到根目录下。修改www-data用户对于html文件夹的权限（php默认的用户貌似就是www-data）

接下来我们再看看flag.sh文件的内容

```
sed -i "s/flag{y0u_4re_mas7er_1n_v1ew_sourc4}/$FLAG/" /var/www/html/index.php
export FLAG=not_flag
FLAG=not_flag

rm -f /flag.sh
```

第一句sed命令是什么意思呢，sed 是 stream editor 的缩写，中文称之为“流编辑器”。第一句话实际起到的作用就是把index.php中的flag替换为容器内的环境变量$FLAG，为之后上传平台时实现动态flag实现可能。

之后的export FLAG=not_flag 和FLAG=not_flag嘛…说实话没有看懂。

最后一句 rm -f /flag.sh就是把此脚本删掉，删除痕迹。虽然我也不知道这个脚本是那一步执行的，难道是在Dockerfile里把它cp到根目录的时候它就会自动执行？可能是这样。

最后还是放以下docker-compose.yml的内容吧。

```
# F12
version: "2"

services:

  web:
    build: .
    image: ctftraining/base_image_nginx_mysql_php_73
    restart: always
    ports:
      - "0.0.0.0:8300:80"
    environment:
      - FLAG=flag{y0u_4re_mas7er_1n_v1ew_sourc4}
```

有了docker-compose.yml ，我们就可以用 docker-compose up -d来快速构建容器，因为在文件里已经标明了build ports和environment 相当与代替了

```
docker build -t f12 . && docker run -d --name -p 8300:80 -e FLAG=flag{y0u_4re_mas7er_1n_v1ew_sourc4} --rm f12
```

这两个命令。然而到最后在CTFd新建题目的时候是不会用到这个文件的，因为平台会绑定自己的端口并且设置自己的环境变量，那个环境变量就是动态flag。

因为题目总是需要不断的调试，我总是用&&符号来连接4个主要命令进行快速重新构建和启动，假设现在已经有一个f12容器正在运行了，我们现在发现了一点错误，想要修改后重启，只需要执行以下指令

```
docker stop f12 && docker rmi f12 && docker build -t f12 . && docker run -d --name=f12 -p 8300:80 --rm f12
```

第一个命令，即关闭容器，因为之前运行的时候加了–rm，所以容器会在关闭后自动删除，如果之前忘记加–rm了的话，我们在stop之后还需要手动删除容器，即docker rm f12

之后的rmi表示删除镜像。

然后build重新构建新的镜像，-t表示这个镜像的名字是f12 那个点 . 不要忘记，它表示当前文件夹，在构建容器的时候会把当前文件全部纳入其中。

最后的run就是启动容器了，-d 表示在后台运行， --name表示容器的名字， -p表示端口映射，–rm之前说过， 最后的f12表示这个容器时用的哪个镜像。

然后我们在浏览器中访问对应端口就行啦

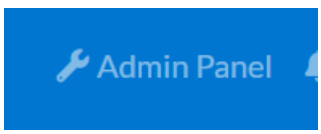
这里的192.168.40.129表示虚拟机在vmware内网中的ip

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.40.129  netmask 255.255.255.0  broadcast 192.168.40.2
        inet6 fe80::20c:29ff:fe08:2a95  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:08:2a:95  txqueuelen 1000  (Ethernet)
        RX packets 7389  bytes 1077923 (1.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 7468  bytes 4396303 (4.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

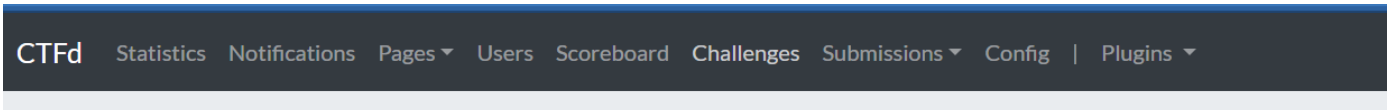
然后8300就表示我们之前docker run 容器的时候绑定的端口，相当于把容器内部的80端口映射到了虚拟机的8300上，又因为在虚拟机的网路是主机提供的，相当与主机是一个路由器，所以在主机中是可以访问虚拟机的网络的，在一层层套娃下，我们在物理机的edge浏览器中访问到了虚拟机里docker容器里的php文件。

最后我们来讲一下怎么把题目上传到CTFd平台上。

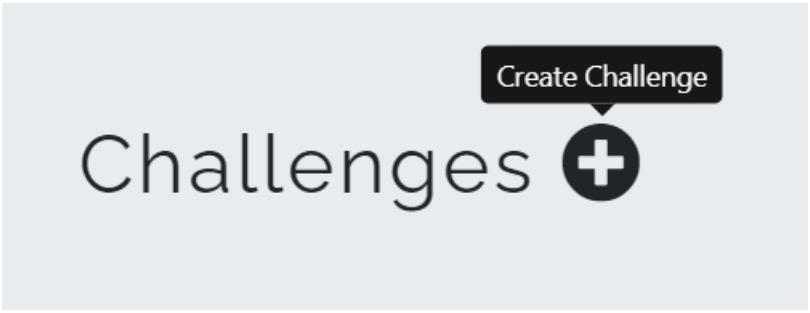
点击Admin Panel按钮，进入管理员模块



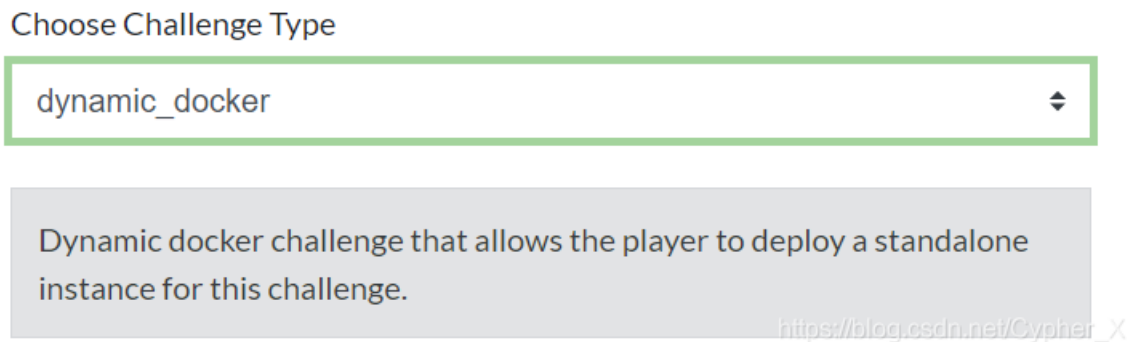
然后点击challenge按钮



进入后点击加号

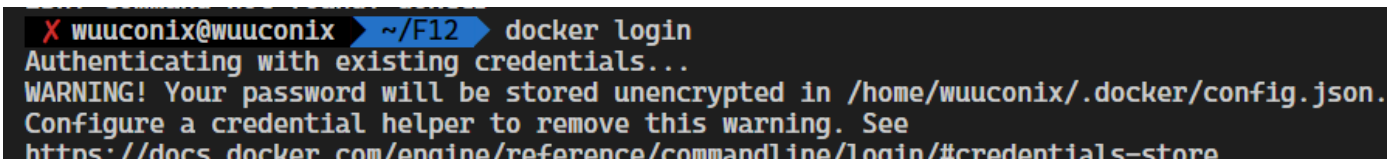


选择动态靶机。



之后一些设置很简单，不细讲，但是其中一个设置很关键，就是选择镜像

那如何把本地的f12镜像上传到靶机上呢？我们需要在Docker Hub上面注册一个账号，然后在终端里登录。



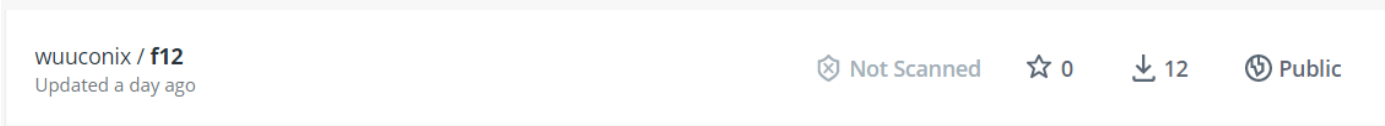
```
Login Succeeded
```

登录之后我们就可以用docker push来上传我们的镜像了，但是之前要重新构建一下，并且在镜像之前加上自己的用户名，类似这样

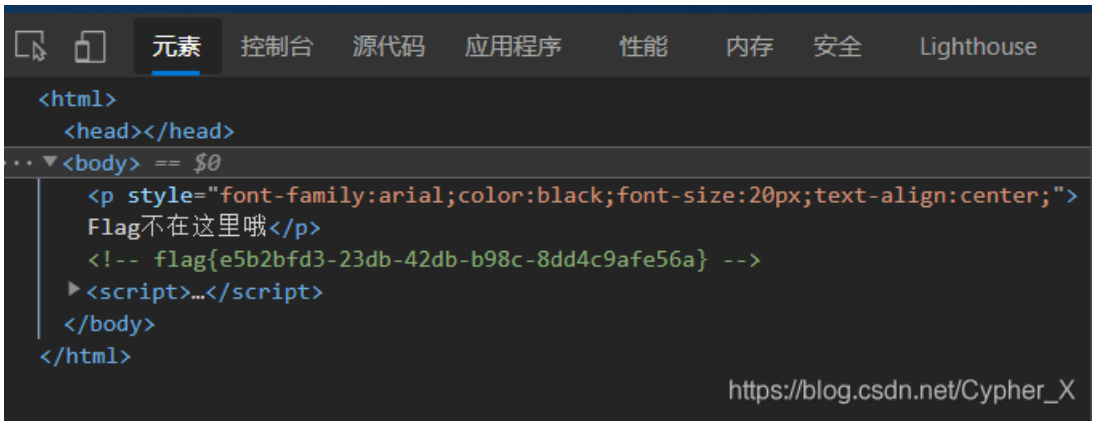
```
wuuconix@wuuconix ~/F12 docker build -t wuuconix/f12 ./ && docker push wuuconix/f12
```

不知道什么原因，我上传一直失败，十次里失败九次（

如果你上传成功了，你就会在Docker hub里看到你上传的镜像。



之后把这个镜像名填写到CTFd平台里，平台就会自动根据这个镜像来构造动态靶机，并且会在环境变量里添加$FLAG，再配合上我们自己写的flag.sh文件，实现了动态flag的功能。



本人菜鸡一只，如果有错误，希望大佬指正。