

PbootCMS 3.0.4 SQL 注入漏洞复现

描述

PbootCMS 是全新内核且永久开源免费的 PHP 企业网站开发建设统，是一套高效、简洁、强悍的可免费商用的 PHP CMS 源码，但存在 SQL 注入漏洞，攻击者可构造恶意语句进行获取敏感数据。

影响范围

PbootCMS 3.0.4

FOFA

app="PBOOTCMS"

源码分析

漏洞代码位置：

```
apps\home\controller\ParserController.php
```

```

385 * 调用本方法时与前面条件使用AND连接
386 * @param boolean $fuzzy
387 * 条件是否为模糊匹配, 即in匹配
388 * @return \core\basic\Model
389 */
390 final public function where($where, $inConnect = 'AND', $outConnect = 'AND', $fuzzy = false)
391 {
392     if (! $where) {
393         return $this;
394     }
395     if (isset($this->sql['where']) && $this->sql['where']) {
396         $this->sql['where'] .= ' ' . $outConnect . '(';
397     } else {
398         $this->sql['where'] = 'WHERE(';
399     }
400     if (is_array($where)) {
401         $where_string = '';
402         $flag = false;
403         foreach ($where as $key => $value) {
404             if ($flag) { // 条件之间内部AND连接
405                 $where_string .= ' ' . $inConnect . ' ';
406             } else {
407                 $flag = true;
408             }
409             if (! is_int($key)) {
410                 if ($fuzzy) {
411                     $where_string .= $key . " like '%" . $value . "%' ";
412                 } else {
413                     $where_string .= $key . "=" . $value . " ";
414                 }
415             } else {
416                 $where_string .= $value;
417             }
418         }
419         $this->sql['where'] .= $where_string . ')';
420     } else {
421         $this->sql['where'] .= $where . ')';
422     }
423     return $this;
424 }
425

```

Qingy之安全

当传递的参数 \$where 是一个数组时就遍历数组, 当 \$where 是一个索引数组时则: \$where_string.= \$value。

接下来找到“\$where”函数中要传递的代码为索引数组时的代码:

pbootcms\static\backup\sql\0cb2353f8ea80b398754308f15d1121e_20200705235534_pbootcms.sql

在“parserSearchLabel()”方法中, 传入的数据被分配到变量“\$receive”进行遍历, “\$key”被带入“request()”进行过滤。

```

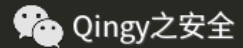
// 解析内容搜索结果标签
public function parserSearchLabel($content)
{
    $pattern = '/\{pboot:search(\s+[^}]+)?\}([\s\S]*)\{\pboot:search\}/';
    $pattern2 = '/\{search:([\w]+)(\s+[^}]+)?\}/';
    if (preg_match_all($pattern, $content, $matches)) {
        $count = count($matches[0]);
        $field = request('field');
        if (! preg_match('/^[w\w|\s]+$/', $field)) {
            $field = '';
        }
        $keyword = request('keyword', 'vars');
        $scode = request('scode');
        $start = 1;
    }
}

```

```

if (! preg_match('/^\[w\s]+\$/ ', $scode)) {
    $scode = '';
}

```



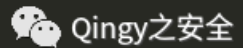
```

// 数据接收
if ($_POST) {
    $receive = $_POST;
} else {
    $receive = $_GET;
}

foreach ($receive as $key => $value) {
    if (! $value = request($key, 'vars')) {
        if ($key == 'title') {
            $key = 'a.title';
        }
        if (preg_match('/^\[w\s]+\$/ ', $key)) { // 带有违规字符时不带入查询
            $where3[$key] = $value;
        }
    }
}

// 去除特殊键值

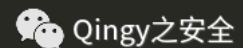
```



```

528 function request($name, $type = null, $require = false, $vartext = null, $default = null)
529 {
530     if (isset($_POST[$name])) {
531         $d_source = 'post';
532     } else {
533         $d_source = 'get';
534     }
535     $condition = array(
536         'd_source' => $d_source,
537         'd_type' => $type,
538         'd_require' => $require,
539         $name => $vartext,
540         'd_default' => $default
541     );
542     return filter($name, $condition);
543 }
544
545

```

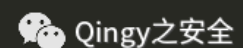


```

function filter($varname, $condition)
{
    // 变量名称文本
    if (array_key_exists($varname, $condition) && $condition[$varname]) {
        $vartext = $condition[$varname];
    } else {
        $vartext = $varname;
    }

    // 数据源
    if (array_key_exists('d_source', $condition)) {
        switch ($condition['d_source']) {
            case 'post':
                $data = @$_POST[$varname];
                break;
            case 'get':
                $data = @$_GET[$varname];
                break;
            case 'cookie':
                $data = @$_COOKIE[$varname];
                break;
        }
    }
}

```



```
// 去空格
if (is_string($data))
    $data = trim($data);
} else {
    $data = $varname; // 没有数据源指定时直接按照字符串过滤处理
}

// 数据为空时,进行是否允许空检测
if (! $data && array_key_exists('d none', $condition) && $condition['d none'] === false)
```

Qingy之安全

```
// 数据类型检测
if (array_key_exists('d_type', $condition)) {
    switch ($condition['d_type']) {
        case 'int':
            if (! preg_match('/^[0-9]+$/', $data)) {
                $err = '必须为整数!';
            }
            break;
        case 'float':
            if (! is_float($data)) {
```

Qingy之安全

```
case 'vars':
    if (! preg_match('/^[\\x{4e00}-\\x{9fa5}\\w\\-\\.\\,\\s]+$/', $data)) {
        $err = '只能包含中文、字母、数字、横线、点、逗号、空格!';
    }
```

Qingy之安全

```
if (is_string($data)) {
    $data = trim($data); // 去空格
    $data = preg_replace_r('/(x3c)|(x3e)/', '', $data); // 去十六进制括号
    $data = preg_replace_r('/pboot:if/i', 'pboot@if', $data); // 过滤插入cms条件语句
    $data = preg_replace_r('/GET\\[/i', 'GET@[', $data);
    $data = preg_replace_r('/POST\\[/i', 'POST@[', $data);
}

// 销毁错误
unset($err);

// 返回数据
return escape_string($data);
}
```

Qingy之安全

```
// 获取传入数据，支持字符串、数组、对象
function escape_string($string)
{
    if (! $string)
        return $string;
    if (is_array($string)) { // 数组处理
        foreach ($string as $key => $value) {
            $string[$key] = escape_string($value);
        }
    } elseif (is_object($string)) { // 对象处理
        foreach ($string as $key => $value) {
            $string->$key = escape_string($value);
        }
    } else { // 字符串处理
        $string = htmlspecialchars(trim($string), ENT_QUOTES, 'UTF-8');
        $string = addslashes($string);
    }
    return $string;
}
```

Qingy之安全

通过上述方法传入索引数组的值只能包含中文、字母、数字、水平线、点、逗号和空格！它由“htmlspecialchars()”和“addslashes()”编码。最后，它被传递到“\$where3”。

```
3084
3085 // 读取数据
3086 if ($page) {
3087     if (isset($paging)) {
3088         error('请不要在一个页面使用多个具有分页的列表，您可将多余的使用page=0关闭分页！');
3089     } else {
3090         $paging = true;
3091         $data = $this->model->getLists($scode, $num, $order, $where1, $where2, $where3, $fuzzy, $start, $lfield, $lg);
3092     }
3093 } else {
3094     $data = $this->model->getList($scode, $num, $order, $where1, $where2, $where3, $fuzzy, $start, $lfield, $lg);
3095 }
```

Qingy之安全

```
// 列表内容 带分页 不分语言 兼容跨语言
public function getLists($scode, $num, $order, $filter = array(), $tags = array(), $select = array(), $fuzzy = true, $start = 1, $lfield = null, $lg = null)
{
    $ext_table = false;
    if ($lfield) {
        $lfield .= ',id,outlink,type,scode,sortfilename,filename,urlname'; // 附加必须字段
        $fields = explode(',', $lfield);
        $fields = array_unique($fields); // 去重
        foreach ($fields as $key => $value) {
            if (strpos($value, 'ext_') === 0) {
                $ext_table = true;
                $fields[$key] = 'e.' . $value;
            } elseif ($value == 'content') {

```

Qingy之安全

```

    }

    // 筛选条件支持模糊匹配
    return parent::table('ay_content a')->field($fields)
        ->where($scode_arr, 'OR')
        ->where($where)
        ->where($select, 'AND', 'AND', $fuzzy)
        ->where($filter, 'OR')
        ->where($tags, 'OR')
        ->join($join)
        ->order($order)

```

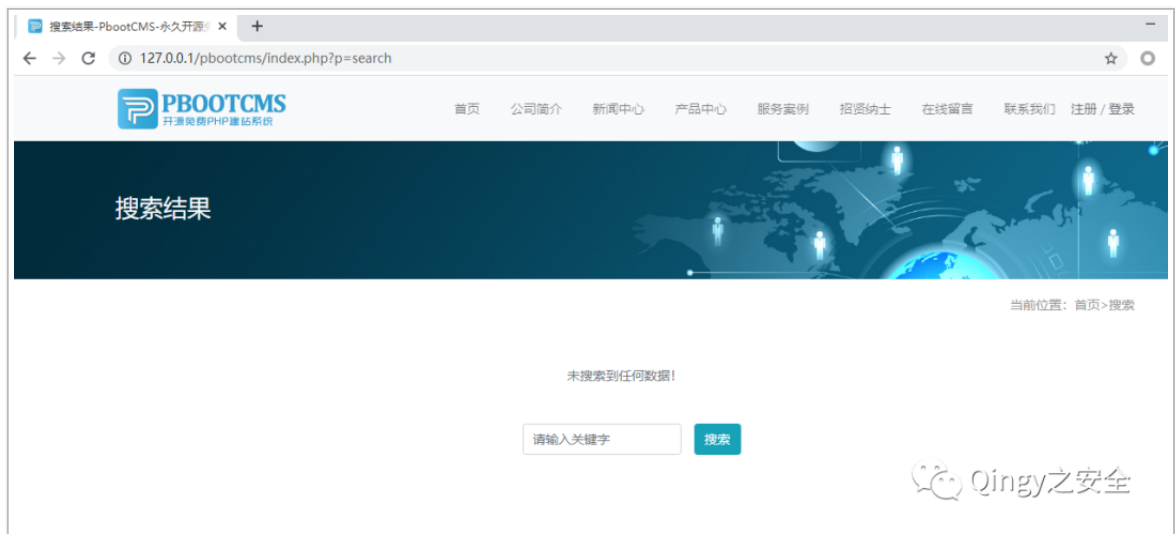


“getlists()”中的“\$where3”是可控的，它将以“and”的形式进入语句，所以最终造成了 SQL 注入。

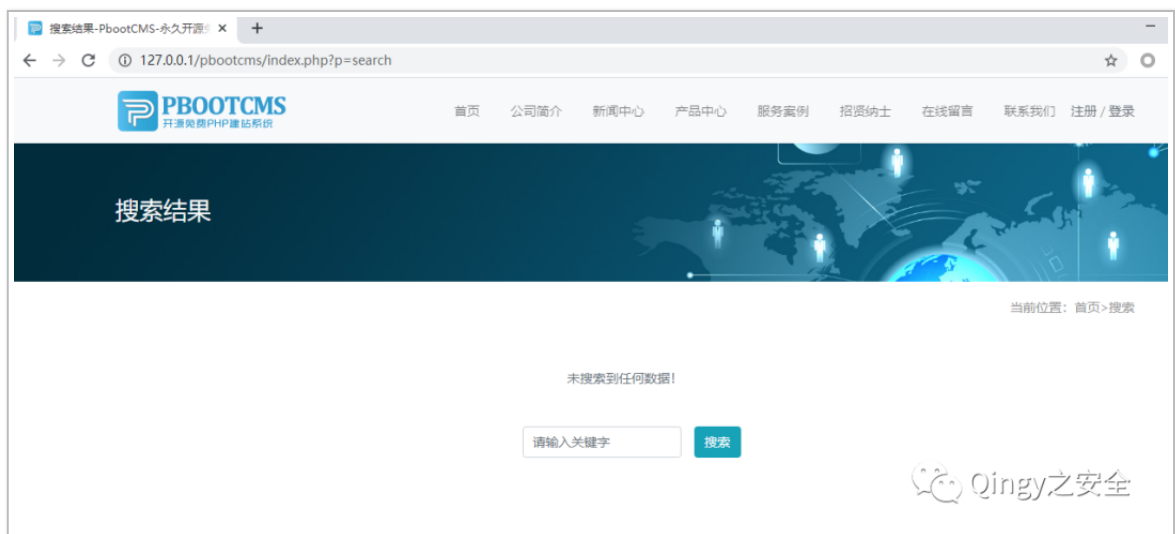
本地复现

默认数据库是 sqlite。为了测试方便，我们需要用 mysql 数据库替换默认数据库。
mysql 数据库目录：

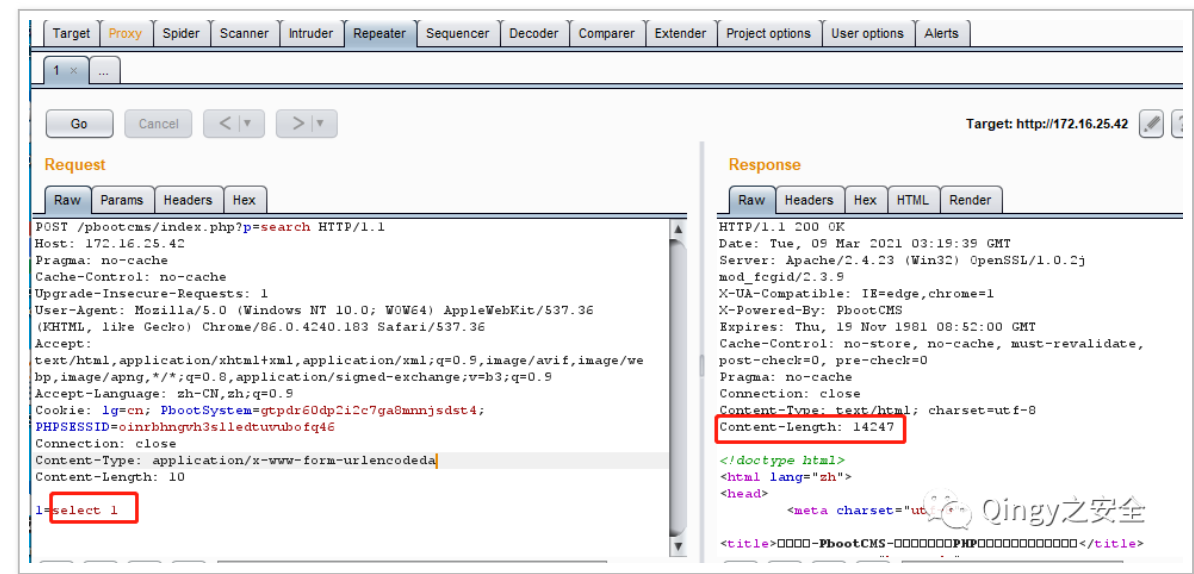
```
pbootcms\static\backup\sql\0cb2353f8ea80b398754308f15d1121e_20200705235534_pbootcms.sql
```



接下来，我们以 POST 的形式发送索引数组，还记得源码里数组中的值要以“and”的形式进入“where”条件：

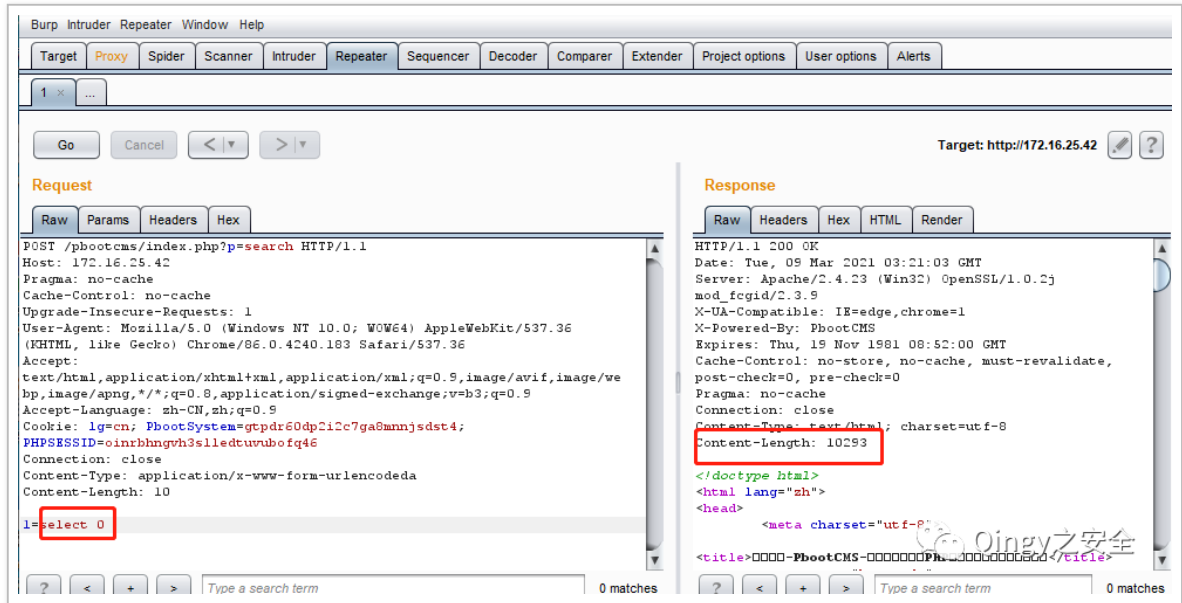
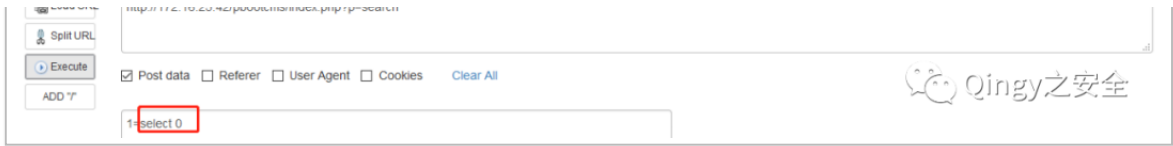


当条件为真时：



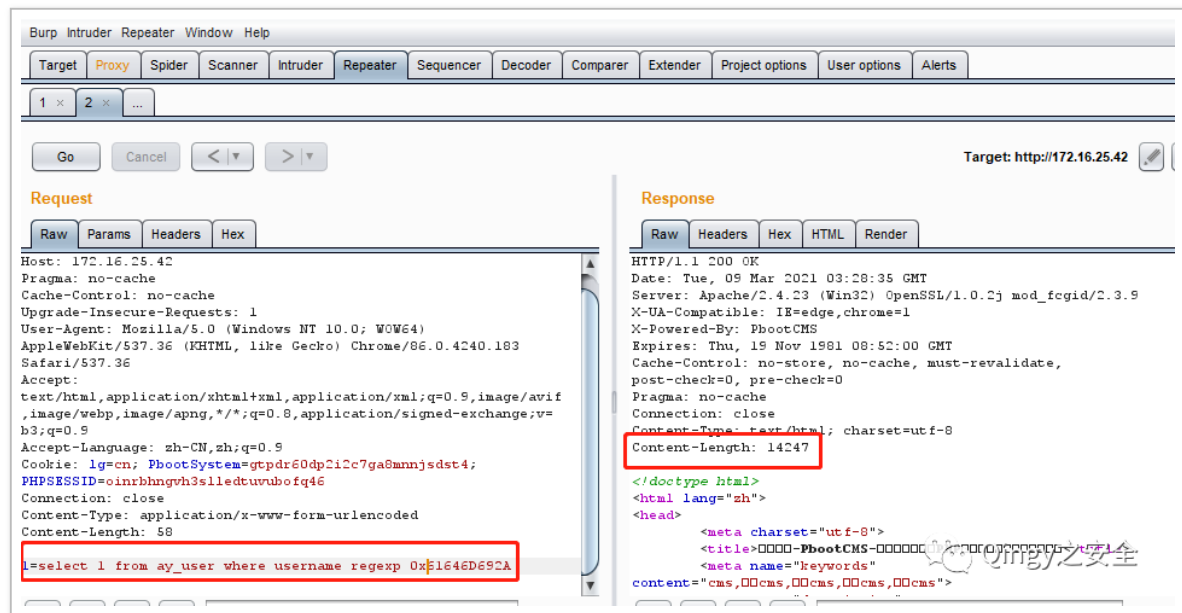
当条件为假时：





有效载荷：由于数据经过过滤，因此只能使用“正则表达式”进行常规匹配。例如：“用户名 = 管理员”可以表示为“用户名 regexp 0x5E612E2A”，其中“5E612E2A”是“^ a”的十六进制代码。





就可以获得管理员的账号密码了。