PbootCMS漏洞合集之审计全过程之二-了解系统参数与底层过滤情况

phpoop / 2018-12-11 06:20:00 / 浏览数 5919 技术文章 技术文章 顶(0) 踩(0)

0x05 清楚地知道系统参数过滤情况

0x05.1 原生 GET, POST, REQUEST

使用原生GET, POST, REQUEST变量是完全不过滤的

测试方法:

最简单的就是这样了

顺便找处系统中可外部访问的地方

如图:

```
MessageController.php - PbootCMS-V1.2.1 - Visu
文件(E) 编辑(E) 选择(S) 查看(V) 转到(G) 调试(D) 任务(I) 帮助(H)
                                                              MessageController.php x
      ▲ 打开的编辑器
Q
         💏 index.php

■ PBOOTCMS-V1.2.1

         AdminController.php
          AdminModel.php
ij.
          ApiController.php
          function.php
                                                      namespace app\home\controller;
\otimes
          * HomeController.php
                                                      use app\home\model\ParserModel;
          💝 route.php
                                                      use core\basic\Controller;
          m version.php
                                                      class MessageController extends Controller
           AboutController.php
                                                          protected $model;
           ContentController.php
           ToController.php
                                                          public function __construct()
            FormController.php
            IndexController.php
       ListController.php
            MessageController.php
            ParserController.php
                                                          public function add()
            SearchController.php
            SitemapController.php
                                                             var_dump($_GET);
var_dump($_POST);
                                                                                                     ▼ 先知社区
        ▶ config
                                                              var_dump($_REQUEST);
```



 $string = preg_replace('/(0x7e)|(0x27)|(0x22)|(updatexml)|(extractvalue)|(name_const)|(concat)/i', '', string);$

} else { // ■■■■
 if (\$dropStr) {

```
}
     $string = htmlspecialchars(trim($string), ENT_QUOTES, 'UTF-8');
     $string = addslashes($string);
  }
  return $string;
}
  可以看得到所有传过来的内容都会先过一个正则匹配过滤
  会将 0x7e, 0x27, 0x22, updatexml, extractvalue, name_const, concat 将其替换为"
  然后在进行
  htmlspecialchars 函数的html实体编码
  addslashes 函数转义。
1. IIIII value
  2. key ■■■■■■
  那么看到这里对于系统的初步情况其实就已经很明确了
0x06 查看系统DB类,了解数据库底层运行方式
在挖sql注入的时候,我最喜欢的就是先看这个系统的底层了。
会对挖洞有很大的帮助,看的时候并不需要全看懂,只需要对一些关键的地方看个大概即可
路径: PbootCMS-V1.2.1\core\basic\Model.php
看了一下以后发现,这里没什么好讲的,因为里面全是字符串拼接。
例如: where方法
   *
   * @param mixed $where
              ■■■■■■"id<1", "name like %1",
              ■■■■array('username'=>'xie',"realname like '%■%'")
   * @param string $inConnect
              $where AND
   * @param string $outConnect
              AND
   * @param boolean $fuzzy
             in
   * @return \core\basic\Model
  final public function where($where, $inConnect = 'AND', $outConnect = 'AND', $fuzzy = false)
  {
     if (! $where) {
         return $this;
     if (isset($this->sql['where']) && $this->sql['where']) {
         $this->sql['where'] .= ' ' . $outConnect . '(';
     } else {
         $this->sql['where'] = 'WHERE(';
     if (is_array($where)) {
         $where_string = '';
         $flag = false;
         foreach ($where as $key => $value) {
            if ($flag) { // ■■■■■■AND■■
               $where_string .= ' ' . $inConnect . ' ';
            } else {
               $flag = true;
            }
            if (! is_int(\$key)) {
               if ($fuzzy) {
                   $where_string .= $key . " like '%" . $value . "%' ";
               } else {
```

```
$where_string .= $value;
          }
          $this->sql['where'] .= $where_string . ')';
      } else {
         $this->sql['where'] .= $where . ')';
      return $this;
  }
初步结论:整个db 类的底层都是类似的字符串拼接,所以(≥□≤)/只要能够带入,或是\那么就可以确定是有注入的了
注意点:在查看的时候发现了insert方法的注释,也是需要注意的,这里我把代码贴一下
   *
   * @param array $data
               _____
               ■■■■■array(
               \verb"array"('username'=>"xsh",'sex'=>'\lefta')",
               array('username'=>"gmx",'sex'=>'■')
               )
   * @param boolean $batch
              THE STATE OF THE STATE
   * @return boolean|boolean|array
  final public function insert(array $data = array(), $batch = true)
   {
      // BEEFFERdata
      if (! $data && isset($this->sql['data'])) {
          return $this->insert($this->sql['data']);
      if (is_array($data)) {
          if (! $data)
             return;
          if (count($data) == count($data, 1)) { // \blacksquare
             $keys = '';
             $values = '';
             foreach ($data as $key => $value) {
                 if (! is_numeric($key)) {
                    $keys .= "`" . $key . "`,";
                     $values .= "'" . $value . "',";
             if ($this->autoTimestamp || (isset($this->sql['auto_time']) && $this->sql['auto_time'] == true)) {
                 $keys .= "`" . $this->createTimeField . "`,`" . $this->updateTimeField . "`,";
                 if ($this->intTimeFormat) {
                     $values .= "'" . time() . "','" . time() . "',";
                     $values .= "'" . date('Y-m-d H:i:s') . "','" . date('Y-m-d H:i:s') . "',";
             $this->sql['field'] = '(' . substr($keys, 0, - 1) . ')';
              } elseif (isset($this->sql['field']) && $this->sql['field']) {
                 $this->sql['field'] = "({$this->sql['field']})";
             $this->sql['value'] = "(" . substr($values, 0, - 1) . ")";
              $sql = $this->buildSql($this->insertSql);
          } else { // ■■■■
              if ($batch) { // ■■■■■■
                 $key_string = '';
                 $value_string = '';
                 $flag = false;
```

\$where_string .= \$key . "='" . \$value . "' ";

} else {

```
foreach ($data as $keys => $value) {
                if (! $flag) {
                    $value_string .= ' SELECT ';
                } else {
                    $value_string .= ' UNION All SELECT ';
                foreach ($value as $key2 => $value2) {
                    //
                    if (! $flag && ! is_numeric($key2)) {
                        $key_string .= "`" . $key2 . "`,";
                    $value_string .= "'" . $value2 . "',";
                }
                $flag = true;
                 if (\$this->autoTimestamp \mid | (isset(\$this->sql['auto\_time']) \&\& \$this->sql['auto\_time'] == true)) \ \{ (this->autoTimestamp \mid | (this->sql['auto\_time']) \&\& \$this->sql['auto\_time'] == true) \} 
                    if ($this->intTimeFormat) {
                        $value_string .= "'" . time() . "','" . time() . "',";
                    } else {
                        $value_string .= "'" . date('Y-m-d H:i:s') . "','" . date('Y-m-d H:i:s') . "',";
                }
                $value_string = substr($value_string, 0, - 1);
            }
            if ($this->autoTimestamp || (isset($this->sql['auto_time']) && $this->sql['auto_time'] == true)) {
                $key\_string .= "`" . $this->createTimeField . "`,`" . $this->updateTimeField . "`,";
            $this->sql['field'] = '(' . substr($key_string, 0, - 1) . ')';
            } elseif (isset(\frac{\pi}{\pi}) elseif (isset(\frac{\pi}{\pi}) elseif (isset(\frac{\pi}{\pi}) elseif (isset(\frac{\pi}{\pi})) {
                \frac{\pi}{\pi} = \frac{\pi}{\pi} ({\pi - sql['field']}) 
            $this->sql['value'] = $value_string;
            $sql = $this->buildSql($this->insertMultSql);
            if (get_db_type() == 'mysql') {
                $max_allowed_packet = $this->getDb()->one('SELECT @@global.max_allowed_packet', 2);
            } else {
                if (strlen(\$sql) > \$max\_allowed\_packet) { // }
                return $this->insert($data, false);
            }
        } else { // ■■■■■■■
            foreach ($data as $keys => $value) {
               $result = $this->insert($value);
            return $result;
        }
    }
} elseif ($this->sql['from']) {
    if (isset(\frac{\pi}{\sin^2\beta}) && \frac{\pi}{\sin^2\beta}) { //
        \frac{\pi}{\pi} = \frac{\pi}{\pi} ({\pi - sql['field']}) 
    $sql = $this->buildSql($this->insertFromSql);
} else {
   return;
return $this->getDb()->amd($sql);
```

简单的说,就是insert 方法支持二维数组,当使用二维数组时表示批量插入。 所以如果我们可以插入二维数组并且可以控制key那么我们就可以注入了。

}

```
下图为触发点查看:
                       if ($batch) { // 批量一次性插入
                           $key_string = '';
                           $value_string = '';
                          $flag = false;
foreach ($data as $keys => $value) {
                               if (! $flag) {
                                   $value_string .= ' SELECT ';
                               } else {
                                   $value_string .= ' UNION All SELECT ';
1158
                               foreach ($value as [$key2]=> $value2) {
                                   if (! $flag && ! is_numeric($key2)) {
                                       $key_string .= "`" . $key2 . "`,";
                                   $value_string .= "'" . $value2 . "',";
                               $flag = true;
                               if ($this->autoTimestamp || (isset($this->sql['auto_time']) && $this->sql['auto_time'] == true)) {
                                   if ($this->intTimeFormat) {
                                       $value_string .= "'" . time() . "','" . time() . "',";
                                   } else {
                                       $value_string .= "'" . date('Y-m-d H:i:s') . "','" . date('Y-m-d H:i:s') . "',";
                               $value_string = substr($value_string, 0, - 1);
```

0x07 系统情况初步集合

经过0x05.1与0x05.2 还有0x06组合下来我们其实可以确定了一些基本漏洞了

5.1 反应给我的情况

首先是xss漏洞

储蓄xss

如果是使用了 5.1 的外部变量并且入库的时候没有转义的话,那么就会产生xss

反射xss

反不动 只要 GET 请求中出现了 A-Z a-z 0-9 之外的数,就会直接报错

然后是sql注入漏洞

如果是使用了 5.1 的外部变量并且入库的时候没有转义的话,那么就会产生sql注入

5.2 反应给我的情况

首先是xss漏洞

储蓄xss

如果是使用了 5.2 的外部变量入库的,想找xss 那么就要看使用了 htmlspecialchars_decode 函数的地方,否则的话就只能查看类似这种点 variate 可控的情况

反射xss

射不动 只要 GET 请求中出现了 A-Z a-z 0-9 之外的数,就会直接报错

然后是sql注入漏洞

key 没有过滤所以如果我能够控制 key 进入sql的话,那么就基本上百分之90有sql注入了

6.0 给我的情况

底层全是字符串拼接,只要能够引入'或是\就可以造成注入

- 5.1 只要我们可以控制就有注入
- 5.2 因为所有的 value 都会进行 htmlspecialchars 函数的html实体编码 addslashes 函数转义所以利用value 注入不现实,那么我们找注入的关键点就是查看key了
- 嗯,可以看到这就很明了了,我的审计收集工作也正式算是做完了。

接下来就是正式审计了。

不过其实到这一步,基本上在看两眼搜索一下打打 debug 就可以确定漏洞了。

点击收藏 | 0 关注 | 1

上一篇:新型网络钓鱼活动事件分析 下一篇:浅析PHP正则表达式的利用技巧

1. 3 条回复



SetObject 2018-12-11 16:38:47

跟表哥学思路,坐等表哥更Xin!

0 回复Ta



Thinker 2018-12-15 22:50:24

老哥你这个是什么插件呀?

0 回复Ta



phpoop 2018-12-26 15:59:27

@z66fr**** vs code

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS <u>关于社区</u> 友情链接 <u>社区小黑板</u>