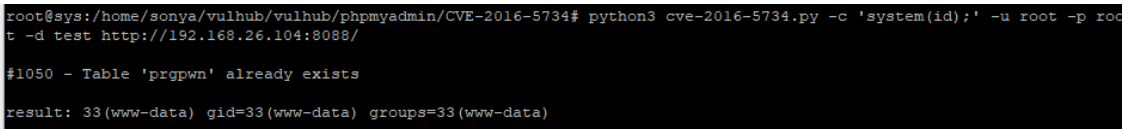
漏洞攻防实训

- http://vulhub.org/#/environments/phpmyadmin/CVE-2016-5734/
- 靶场环境构建
  - 使用 Vulhub 一键搭建漏洞测试靶场
    - 8080端口占用 修改docker-compose.yml 中的 8080:80 -> 8088:80



  - （可选）自己也可以基于 Docker 编写新的漏洞测试靶场
- 漏洞利用实战
  - 使用工具手工或自动化完成
    - https://www.exploit-db.com/exploits/40185/ 结果如上 对exp的注释阅读
  - （可选）自己编写自动化 **exploit** 脚本
- 漏洞原理分析
  - 漏洞触发代码定位：
    - 根据漏洞触发代码所在位置**tbl_find_replace.php**中的代码可知替换使用了**PMA_TableSearch**类的**getReplacePreview**方法

```
<?php
  $table_search = new PMA_TableSearch($db, $table,
"replace");
  if (isset($_POST['find'])) {
    $preview = $table_search->getReplacePreview(
        $_POST['columnIndex'],
        $_POST['find'],
        $_POST['replaceWith'],
        $_POST['useRegex'],
        $connectionCharSet
    );
    $response->addJSON('preview', $preview);
    exit;
}
  ?>
```

- 查看**TableSearch.class.php**中的**getReplacePreview**方法 可知在使用正则的情况下 会先调用**_getRegexReplaceRows**方法

```
function getReplacePreview($columnIndex, $find,
$replaceWith, $useRegex,
    $charSet
) {
    $column = $this->_columnNames[$columnIndex];
    if ($useRegex) {
        $result = $this->_getRegexReplaceRows(
            $columnIndex, $find, $replaceWith, $charSet
        );
```

- 查看同文件下的**_getRegexReplaceRows** 方法，发现这个方法会先将数据库查询返回结果使用**preg_replace** 函数进行替换，这就定位到了漏洞触发位置。

```
function _getRegexReplaceRows($columnIndex, $find,
$replaceWith, $charSet)
    {
        ....
        if (is_array($result)) {
            foreach ($result as $index=>$row) {
                $result[$index][1] = preg_replace(
                    "/" . $find . "/",
                    $replaceWith,
                    $row[0]
                );
            }
        }
        return $result;
    }
```

- 漏洞原理
  - 漏洞主要是因为使用了不安全函数 preg_replace(http://php.net/manual/zh/function.preg-replace.php)

    ```
    mixed preg_replace ( mixed $pattern , mixed
    $replacement , mixed $subject [, int $limit = -1 [, int
    &$count ]] )
    ```

    在这个函数中PHP <=5.5.0版本中 pattern 参数支持模式修饰符e，有参数e的时候可以将replacement参数作为php代码执行，将执行结果作为替换字符串。
  - 如 官方文档中给出的例子 (http://php.net/manual/zh/reference.pcre.pattern.modifiers.php)

```php
<?php
$html = $_POST['html'];
// uppercase headings
$html = preg_replace(
    '(<h([1-6])>(.*?)</h\1>)e',
    '"<h$1>" . strtoupper("$2") . "</h$1>"',
    $html
);
```

以上示例代码能够被这样的字符串利用：`<h1>{${eval($_GET[php_code])}}</h1>`。 这能让攻击者执行他们想要的 PHP 代码，几乎完全渗透进服务器。

- 按照 CWE 进行漏洞成因归类