Some writeups for Hackergame

透明的文件

这题拿到题目的文件就是一大串不知明的字符。如果对ANSI不熟悉的话,很难入手。好在本人接触过一些python 命令行输出样式的写法。所以经过简单的learning。最终的做法为:在每个] 前加上 \033 然后将空格全替换成字符,例如'm'。最后flag会长这样。



所以这题玩的是figlet这个梗。另外ANSI可以参考下面的链接。

https://segmentfault.com/a/1190000023553724

赛博厨房level2

这题是真的折磨,我想到了好几种思路。第一种思路是构造出一个程序使得机器人学完后的菜单和程序是对应的。但是看了源码,看了好久根本看不懂是怎么加密的。因此我又想了一个思路,如果机器人不能进行清理这个操作,那么我们就可以重复一个操作行为。就是说可以让机器人学0到31食材的拿取方式,这样不论下一个菜单是什么,我们都可以依次执行。所以我就专攻了第二种思路。

我删除了所以和清理有关的函数 clear(),经过测试果然机器人不清理了。此时我很激动,快速写完32个程序。执行"0,1,2,3,4,5"这是我就感到不太妙。菜单没变还是"0,1,2,3,4,5"。执行完之后alert出了success,我暗喜,这不是成功绕过前端验证了吗?但是,确定之后没有flag,提示菜单不对。该死的后端验证,没想到出题人这么阴。所以这题就摆烂了。

官方解答还是 mcfx

minecRaft

开始看这个题目,说是要点亮三盏灯。F12审了一下源码。可以就判断出思路了。

```
if(cinput.length>=32){
    let tbool=gyflagh(cinput.join(''));
    if(tbool) {
        pressplateList[65].TurnOn_redstone_lamp();
        content.innerText='Congratulations!!!';
        return;
}
```

可以看一下段,然后追溯 gyflagh 这个函数,发现他在 flag.js 这个文件里。

初步看一眼,文件 js 混淆了。目前还没有能够有效反混淆的工具,但是我们可以选择调试。第一种,直接浏览器调试,但是被出题人debug了。第二中就是本地 nodojs 调。

```
function gyflagh(_0x111955) {
    const _0x50051f = _0x22517d;
    console.log(_0x111955);
    let _0x3b790d = _0x111955[_0x50051f(0x1a8)](_0x50051f(0x1b7));
    console.log(_0x3b790d);
    console.log(_0x50051f(0x1aa));
    if (_0x3b790d === _0x50051f(0x1aa)) return !![];
    return ![];
}
```

主要观察一下这个函数,我们发现只要最后一个if语句成立,我们就可以获得flag。那不妨输出一下 _0x50051f(0x1aa)) 发现是

```
6fbde674819a59bfa12092565b4ca2a7a11dc670c678681daf4afb6704b82f0c
```

这时结合这个题目,发现根本没有输出flag的这一功能的函数,所以题目的大概意思是输入flag然后验证。而这里一大长串的就是加密后的flag。所以思路就是读懂其中的加密算法。过程确实有点痛苦,但是只要一点一点调是可以的。

逐步调试后发现code()这个函数就是加密函数。我们只需将其逆过来就行了。好像也不难+=改成-=,顺序倒一下就好了。

```
function decode(_0x167a71, _0x762113) {
    let _0x412874 = _0x167a71[0x0],
        _0x3f9c14 = _0x167a71[0x1];
    const _0x540f95 = (0x52cfb2de + 0x4b67c6db),
        _0x2bdc23 = _0x540f95 * 0x20;
    let _0x4f8e47 = _0x2bdc23;
    while (_0x4f8e47 != 0x0) {
        _0x3f9c14 -= (_0x412874 << 0x4 ^ _0x412874 >>> 0x5) + _0x412874 ^ _0x4f8e47 + _0x762113[_0x4f8e47 >>> 0xb & 0x3];
        _0x4f8e47 -= _0x540f95;
```

```
104 console.log(Base16ToLong("819a59bf"));
105 console.log(Base16ToLong("a1209256"));
106 console.log(Base16ToLong("5b4ca2a7"));
107 console.log(Base16ToLong("a11dc670"));
108 console.log(Base16ToLong("c678681d"));
109 console.log(Base16ToLong("af4afb67"));
110 console.log(Base16ToLong("04b82f0c"));
111
112 console.log(LongToStr4("10290160461"), LongToStr4("-6991555998"));
console.log(LongToStr4("1817013865"), LongToStr4("-3196959890"));
114 console.log(LongToStr4("-3433073615"), LongToStr4("5896746337"));
console.log(LongToStr4("14852645681"), LongToStr4("-6824562577"));
问题 輸出 终端
-1354040473
79179532
McWe bRE
inMl nCrA
1t_3 a5y_
1cIu op9i
```

反正最后flag就这样出了,转码稍微注意一下就行了。