

EasyBox

发表于 2021-01-04 分类于 [Challenge](#) , [2019](#) , [UNCTF](#) , [Misc](#)
[Challenge](#) | [2019](#) | [UNCTF](#) | [Misc](#) | [EasyBox](#)

[点击此处](#)获得更好的阅读体验

题目考点

- 深度优先搜索算法（英语：Depth-First-Search，简称DFS）是一种用于遍历或搜索树或图的算法。沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都已被探寻过或者在搜寻时结点不满足条件，搜索将回溯到发现节点v的那条边的起始节点。整个进程反复进行直到所有节点都被访问为止。属于盲目搜索,最糟糕的情况算法时间复杂度为 $O(n)$ 。
- Pwntools提供了方便的网络交互编程的接口

解题过程

使用nc连接到靶机开放端口。返回结果如下所示，可以看出是一个数独之类的游戏，但是交互时间很短，只能通过编写脚本来完成：

□

利用dfs（深度优先搜索算法）来编写计算数独空缺数字脚本，根据题目提示，这个数独只需要横向和纵向的数字和为45，并且1-9只能出现一次，部分脚本如下所示：

□

exp.py使用pwntools库负责接收数据和发送数据，solve.py负责将数据整理并利用 [exp.py](#)

```

1 #coding:utf-8
2 from pwn import *
3 import datetime
4 import solve
5 #context.log_level='debug'
6 p=remote('123.206.21.178',10000)
7 shudu=[]
8 p.recvuntil("+---+---+---+---+---")
9 p.recvuntil("|")
10 shudu.append(p.recv(17).strip("\n").split('| '))
11 print shudu
12 p.recvuntil("+---+---+---+---+---")
13 p.recvuntil("|")
14 shudu.append(p.recv(17).strip("\n").split('| '))
15 p.recvuntil("+---+---+---+---+---")
16 p.recvuntil("|")
17 shudu.append(p.recv(17).strip("\n").split('| '))
18 p.recvuntil("+---+---+---+---+---")
19 p.recvuntil("|")
20 shudu.append(p.recv(17).strip("\n").split('| '))
21 p.recvuntil("+---+---+---+---+---")
22 p.recvuntil("|")
23 shudu.append(p.recv(17).strip("\n").split('| '))
24 p.recvuntil("+---+---+---+---+---")
25 p.recvuntil("|")
26 shudu.append(p.recv(17).strip("\n").split('| '))
27 p.recvuntil("+---+---+---+---+---")
28 p.recvuntil("|")
29 shudu.append(p.recv(17).strip("\n").split('| '))
30 p.recvuntil("+---+---+---+---+---")
31 p.recvuntil("|")
32 shudu.append(p.recv(17).strip("\n").split('| '))
33 p.recvuntil("+---+---+---+---+---")
34 p.recvuntil("|")
35 shudu.append(p.recv(17).strip("\n").split('| '))
36 res=[]
37 for i in range(9):
38     res.append(['0' if x==' ' else x for x in shudu[i]])
39 m=''
40 for i in range(9):
41     for j in range(9):
42         res[i][j]=int(res[i][j])
43         if res[i][j]==0:
44             m+=str(i)+'.'+str(j)+' '
45 print res
46 m = m[:-1].split(' ')
47 print m
48 res = solve.start(res)
49 answer=[]
50 for i in range(9):
51     s=''
52     for x in m:
53         if i == int(x[0]):
54             s += str(res[int(x[0])][int(x[2])])+' '
55     answer.append(s[:-1])
56 print answer
57 p.recv()
58 for i in answer:
59     payload = i.replace(' ','')
60     print payload
61     p.sendline(payload)
62 p.recvuntil('')
63 p.interactive()

```

[solve.py](#)

```

1 #coding:utf-8
2 def my_dfs(A):
3     result=[]
4     ALL_SET=set({1,2,3,4,5,6,7,8,9,0})#全集
5     for i in range(9):
6         for j in range(9):
7             if A[i][j]==0:
8                 d={}
9                 set_x=set(A[i][k] for k in range(9))#横向的集合
10                set_y=set(A[k][j] for k in range(9))#纵向的集合
11                set_x_y=set_x|set_y
12                data=list(ALL_SET-set_x_y)#总的集合-横向纵向的集合=可能解的集合
13                d['%s,%s'%(str(i),str(j))]=data
14                result.append(d)
15    return result
16 #判断数独中是否含有未确定的数字
17 def check(A):
18     flag=False
19     for i in range(9):
20         for j in range(9):
21             if A[i][j] is 0:
22                 flag=True
23     #A中有0为True
24     return flag
25 def start(A):
26     count=1
27     while check(A):
28         data=my_dfs(A)
29         for each in data:
30             for index,item in each.items():
31                 if len(item)==1:#如果可能解只有一个,那么该点就是这个解
32                     i,j=index.split(',')
33                     A[int(i)][int(j)]=item[0]#将这个解赋值给该点
34             count+=1
35     return A
36 if __name__=='__main__':
37     A = [
38         [0, 8, 9, 0, 1, 6, 0, 2, 0],
39         [8, 0, 3, 0, 4, 0, 7, 0, 6],
40         [7, 1, 2, 9, 0, 8, 0, 4, 5],
41         [4, 7, 0, 6, 9, 5, 3, 1, 2],
42         [0, 6, 7, 0, 8, 4, 2, 9, 0],
43         [0, 0, 0, 4, 7, 0, 0, 0, 9],
44         [0, 0, 5, 3, 6, 0, 0, 7, 0],
45         [0, 0, 0, 0, 5, 1, 8, 6, 7],
46         [6, 9, 1, 8, 2, 7, 0, 0, 4]
47     ]
48     print start(A)

```

深度优先搜索算法来得出空缺的数字,最后由exp.py发送,结果如下:

□

- 本文作者: CTFHub
- 本文链接: <https://writeup.ctfhub.com/Challenge/2019/UNCTF/Misc/iMHxRBrym7WhsPvwMm6q6s.html>
- 版权声明: 本博客所有文章除特别声明外,均采用 [BY-NC-SA](#) 许可协议。转载请注明出处!

[#Challenge](#) [#Misc](#) [#2019](#) [#UNCTF](#)

[BACON](#)

[Happy_puzzle](#)