2021巅峰极客ichunqiu_baby-maze

原创

PoilZero 已于 2022-01-22 18:03:37 修改 1313 收藏 3

分类专栏： CTF-RE 文章标签： python

于 2021-07-31 22:07:40 首次发布

本文链接：https://blog.csdn.net/qq_43583624/article/details/119282253

2021巅峰极客ichunqiu_baby-maze

题目来源和附件

> Written by Poilzero（blog：poilzero.cn）
>
> - 同步转发到简书和CSDN
>
> 题目来源：
>
> - 2021巅峰极客网络安全技能挑战赛
> - reverse方向
> - baby_maze
>
> 个人解题目录和部分所需程序（含题目附件）：
> https://poil.lanzoui.com/iOpu1s329mj

思路

程序逻辑

- 迷宫，输入WASD进行方向移动，Q是退出。
- 答案就是flag{md(输入的WASD组合)}

摸索尝试

因为找不到地图数据，也想换一个思路做题，于是一开始尝试从sub_54DD7E开始逆推，存在多分枝，放弃。

然后尝试爆破但是遇到subprocess不能实时IO的问题，如果按照我之前的写法进程执行完读取全部缓冲区处理感觉速度爆炸，后发觉逆推回去可控，而且可以用ida python正好练练手。

先试了试手推了28位，后放弃 SDD SSA ASS SSS SDD SSA ASS SSS SSS D 原因是地图太大，无法手动逆推，地图规模预计100x100。

正确思路

后来想到一种方法，觉得可行性比较大就是使用ida python编写脚本代替手推

伪代码如下，因为api还不是很熟可能写了很久，边学边写的，当然也可以写一起，我不太熟所以分开写了

```
marked_block = []
def find_path(now_block):
    if now_block == start_block:
        return
    for x in now_block.xref_block():
        if not(x in marked_block):
            marked_block.append(x)
            find_path(x)

if __name__ == '__main__':
     res_block = FuncBlock(addr=0x054DE35)
     find_path(res_block)
     print(marked_block)
```

EXP

requirement

- IDA Pro 7.0
- IDA Python

得到正确路径

```
# coding:utf-8

def fm(s):
    return hex(s)

# after input 13 S
# sub_42360E()
start_block     = 0x40187C
queue_call_refs = []
def find_path(callee_addr, queue_call):
    # print(hex(callee_addr))
    '''
        0x53cdaeL
        0x53ce65L
        bad circle
    '''
    if callee_addr == start_block:
        print '[+] ' + str(len(queue_call))
        print '[+] Path found:\n' + '-'.join(queue_call)
        print '[+] Path refs:\n' + '-'.join(queue_call_refs)
        return

    # for x in now_block.xref_block():
    for refs_addr in idautils.CodeRefsTo(callee_addr, 0):
        caller_name = GetFunctionName(refs_addr)
        caller_addr = LocByName(caller_name)

        if fm(caller_addr) in queue_call:
            continue

        queue_call.append(fm(caller_addr))
        queue_call_refs.append(fm(refs_addr))
        try:
            find_path(caller_addr, queue_call)
        except BaseException,e:
            print e
        queue_call.remove(fm(caller_addr))
        queue_call_refs.remove(fm(refs_addr))

print '\n'*0x10000
print '[+] start to search'
res_block = 0x054DE35
ls = []
find_path(res_block, ls)
print '[+] end for search'
```

路径转换为WASD

s, s1为上一个脚本输出的结果

```
# coding:utf-8

s = '0x54dd7eL-0x54dcc7L-0x54dc10L-0x5490e9L-0x547f0eL-0x547fc1L-0x548078L-0x543df5L-0x54265eL-0x53e76eL-0x53cfd
7L-0x538e0fL-0x537b79L-0x537ac2L-0x537a0bL-0x5339a9L-0x532212L-0x5322c9L-0x532380L-0x52e287L-0x52c8cbL-0x5289e3L
-0x527303L-0x52340bL-0x521d2bL-0x51de3bL-0x51c5edL-0x51c536L-0x51c47fL-0x51c3c8L-0x51c311L-0x51c25aL-0x51c1a3L-0
x51dd84L-0x521a4bL-0x521994L-0x5218ddL-0x51dccdL-0x51c0ecL-0x518656L-0x5169baL-0x512cfbL-0x51128cL-0x5111d5L-0x5
1111eL-0x511067L-0x510fb4L-0x50d5c5L-0x50b9dcL-0x50b925L-0x50b86eL-0x50b7b7L-0x50b700L-0x50b649L-0x50b592L-0x50b
4dbL-0x50b424L-0x507c6eL-0x505e68L-0x502759L-0x50050dL-0x4fcf5cL-0x4fb209L-0x4fb2c0L-0x4fb377L-0x4f7826L-0x4f5c4
5L-0x4f5b8eL-0x4f5ad7L-0x4f5a20L-0x4f5969L-0x4f58b2L-0x4f57fbL-0x4f1e1cL-0x4f0188L-0x4ec854L-0x4ea772L-0x4ea6bbL
```

```
x4fae76L-0x4fadbfL-0x4fad08L-0x4fac51L-0x4f76b8L-0x4f551fL-0x4f1caeL-0x4eff5fL-0x4ec62fL-0x4ea3e3L-0x4ea32cL-0x4
ea275L-0x4ec578L-0x4efea8L-0x4efdf1L-0x4efd3aL-0x4efc83L-0x4efbccL-0x4ec4c1L-0x4ea103L-0x4ea04cL-0x4e9f99L-0x4e9
ee2L-0x4e9e2bL-0x4e9d74L-0x4e9cc1L-0x4e6cd4L-0x4e4c99L-0x4e4d50L-0x4e4e07L-0x4e19d0L-0x4df119L-0x4df062L-0x4defa
bL-0x4dc2aaL-0x4d97d2L-0x4d9885L-0x4d993cL-0x4d6a0eL-0x4d4370L-0x4d1010L-0x4ceadcL-0x4ceb93L-0x4cbaebL-0x4c972dL
-0x4c9676L-0x4c651bL-0x4c40a6L-0x4c3fefL-0x4c3f3cL-0x4c0e98L-0x4be8b5L-0x4bb3cfL-0x4b917fL-0x4b90c8L-0x4b9015L-0
x4b8f5eL-0x4b8ea7L-0x4b8df0L-0x4b8d39L-0x4b5d60L-0x4b355cL-0x4b0411L-0x4ade2aL-0x4add73L-0x4adcc0L-0x4adc09L-0x4
adb52L-0x4aad8aL-0x4a8410L-0x4a56ffL-0x4a305dL-0x4a2fa6L-0x4a2eefL-0x49ffbdL-0x49d59cL-0x49a88fL-0x497e62L-0x495
214L-0x4927e3L-0x49272cL-0x492675L-0x48f802L-0x48d0b1L-0x48cffaL-0x48cf43L-0x48ce8cL-0x48cdd5L-0x48a45bL-0x48725
5L-0x484b10L-0x481dfbL-0x481d44L-0x481c91L-0x47f264L-0x47c6c5L-0x47c60eL-0x47c557L-0x47c4a0L-0x47c3e9L-0x47c332L
-0x47c27bL-0x47c1c4L-0x47c111L-0x4799c0L-0x476a96L-0x4769dfL-0x476928L-0x474345L-0x471088L-0x470fd1L-0x470f1aL-0
x47428eL-0x476871L-0x4767baL-0x476703L-0x47664cL-0x476599L-0x4741d7L-0x470da8L-0x470cf1L-0x470c3aL-0x470b83L-0x4
70accL-0x46ea99L-0x46b4f4L-0x469363L-0x465f34L-0x465febL-0x4660a2L-0x4638a6L-0x460980L-0x45e22bL-0x45b0d4L-0x458
c57L-0x45599eL-0x453255L-0x4505ebL-0x450534L-0x45047dL-0x45319eL-0x4558e7L-0x458ba0L-0x45b021L-0x45af6aL-0x45aeb
3L-0x45e0bdL-0x46080eL-0x4637efL-0x465e81L-0x465dcaL-0x465d13L-0x4691f5L-0x46b386L-0x46e92bL-0x47095eL-0x474069L
-0x476206L-0x47614fL-0x476098L-0x473fb2L-0x4708a7L-0x46e874L-0x46b2cfL-0x46913eL-0x465c5cL-0x463738L-0x4605e9L-0
x4606a0L-0x460757L-0x45e006L-0x45adfcL-0x458ae9L-0x4556c2L-0x455779L-0x455830L-0x4530e7L-0x4503c6L-0x45030fL-0x4
50258L-0x44de8aL-0x44a9acL-0x44a8f5L-0x44a83eL-0x44a787L-0x44a6d4L-0x44a61dL-0x44a566L-0x44a4afL-0x44a3f8L-0x44a
341L-0x44a28aL-0x448490L-0x444d89L-0x442a86L-0x43f4e1L-0x43f598L-0x43f64fL-0x442b3dL-0x444e40L-0x444ef7L-0x444fa
eL-0x442bf4L-0x43f706L-0x43d575L-0x439d08L-0x439dbfL-0x439e76L-0x437d90L-0x43496dL-0x432441L-0x42f00aL-0x42d08eL
-0x429db9L-0x429e70L-0x429f27L-0x429fdeL-0x42a095L-0x4279f3L-0x4247e5L-0x42472eL-0x424677L-0x4245c0L-0x424509L-0
x42242fL-0x41ed2cL-0x41c8bfL-0x419547L-0x417520L-0x413e15L-0x411d2bL-0x40e565L-0x40c812L-0x408ef6L-0x408e3fL-0x4
08d88L-0x406bf3L-0x402b06L-0x402a4fL-0x40299cL-0x406b3cL-0x408cd1L-0x40c75bL-0x40e340L-0x40e3f7L-0x40e4aeL-0x411
c74L-0x413d5eL-0x413ca7L-0x413bf0L-0x413b39L-0x413a82L-0x417469L-0x4191b4L-0x41926bL-0x419322L-0x4193d9L-0x41949
0L-0x41c808L-0x41ec75L-0x41ebbeL-0x41eb07L-0x41ea50L-0x41e99dL-0x41e8e6L-0x41e82fL-0x4222c1L-0x4240c3L-0x42400cL
-0x423f55L-0x423e9eL-0x423debL-0x42220aL-0x41e6bdL-0x41e606L-0x41e54fL-0x422153L-0x423d34L-0x427717L-0x429693L-0
x4295dcL-0x429525L-0x427660L-0x423c7dL-0x42209cL-0x41e498L-0x41e3e1L-0x41e32eL-0x41c69aL-0x418b45L-0x418bfcL-0x4
18cb3L-0x41718dL-0x413638L-0x411998L-0x40de3bL-0x40c3c8L-0x408666L-0x40871dL-0x4087d4L-0x40c47fL-0x40def2L-0x411
a4fL-0x4136efL-0x4137a2L-0x413859L-0x4172fbL-0x418e25L-0x418edcL-0x418f93L-0x419046L-0x4190fdL-0x4173b2L-0x4139c
bL-0x411bbdL-0x40e11bL-0x40e1d2L-0x40e289L-0x40c6a4L-0x408c1aL-0x408b63L-0x408aacL-0x406a85L-0x402773L-0x4026bcL
-0x402605L-0x40254eL-0x40249bL-0x4023e4L-0x40232dL-0x402276L-0x4021bfL-0x402108L-0x402051L-0x401f9aL-0x401ee3L-0
x401e2cL-0x401d79L-0x401cc2L-0x401c0fL-0x401b58L-0x401aa1L-0x4019eaL-0x401933L-0x4067a9L-0x407ff7L-0x4080aaL-0x4
08161L-0x40c25aL-0x40d93aL-0x40d9f1L-0x40daa8L-0x4116bcL-0x413358L-0x416f68L-0x418869L-0x4187b2L-0x4186fbL-0x418
644L-0x41858dL-0x4184d6L-0x418423L-0x416eb1L-0x413078L-0x41154eL-0x40d7ccL-0x40c0ecL-0x407f40L-0x4066f2L-0x40187
cL'

s2 = '0x54de16L-0x54dd51L-0x54dc96L-0x54917dL-0x547f9eL-0x548063L-0x54811aL-0x543e89L-0x5426f2L-0x53e802L-0x53d0
6bL-0x538ea3L-0x537c11L-0x537b4cL-0x537a91L-0x533a3dL-0x5322a6L-0x53236bL-0x532422L-0x52e31bL-0x52c95fL-0x528a77
L-0x527397L-0x52349fL-0x521dbfL-0x51decfL-0x51c685L-0x51c5c0L-0x51c509L-0x51c452L-0x51c39bL-0x51c2e4L-0x51c22dL-
0x51de00L-0x521ac7L-0x521a1eL-0x521963L-0x51dd61L-0x51c180L-0x5186eaL-0x516a4eL-0x512d8fL-0x511324L-0x51125fL-0x
5111a8L-0x5110f1L-0x51103aL-0x50d659L-0x50ba74L-0x50b9afL-0x50b8f8L-0x50b841L-0x50b78aL-0x50b6d3L-0x50b61cL-0x50
b565L-0x50b4aaL-0x507d02L-0x505efcL-0x5027edL-0x50059dL-0x4fcff0L-0x4fb29dL-0x4fb362L-0x4fb419L-0x4f78baL-0x4f5c
ddL-0x4f5c18L-0x4f5b61L-0x4f5aaaL-0x4f59f3L-0x4f593cL-0x4f5881L-0x4f1eb0L-0x4f0218L-0x4ec8e8L-0x4ea80aL-0x4ea745
L-0x4ea692L-0x4ea5dbL-0x4ea524L-0x4ec762L-0x4f0092L-0x4f1de1L-0x4f5652L-0x4f572fL-0x4f57e6L-0x4f77ebL-0x4fafa9L-
0x4faf00L-0x4fae49L-0x4fad92L-0x4facd7L-0x4f774cL-0x4f55b3L-0x4f1d42L-0x4efff3L-0x4ec6c3L-0x4ea47bL-0x4ea3b6L-0x
4ea2ffL-0x4ec5f4L-0x4eff24L-0x4efe7bL-0x4efdc4L-0x4efd0dL-0x4efc52L-0x4ec555L-0x4ea19bL-0x4ea0d6L-0x4ea023L-0x4e
9f6cL-0x4e9eb5L-0x4e9dfeL-0x4e9d47L-0x4e6d68L-0x4e4d2dL-0x4e4df2L-0x4e4ea9L-0x4e1a64L-0x4df1b1L-0x4df0ecL-0x4df0
31L-0x4dc33eL-0x4d9862L-0x4d9927L-0x4d99deL-0x4d6aa2L-0x4d4404L-0x4d10a4L-0x4ceb70L-0x4cec35L-0x4cbb7fL-0x4c97c5
L-0x4c96fcL-0x4c65afL-0x4c413eL-0x4c4079L-0x4c3fc2L-0x4c0f2cL-0x4be949L-0x4bb463L-0x4b9217L-0x4b9152L-0x4b909fL-
0x4b8fe8L-0x4b8f31L-0x4b8e7aL-0x4b8dbfL-0x4b5df4L-0x4b35f0L-0x4b04a5L-0x4adec2L-0x4addfdL-0x4add4aL-0x4adc93L-0x
4adbd8L-0x4aae1eL-0x4a84a0L-0x4a5793L-0x4a30f5L-0x4a3030L-0x4a2f75L-0x4a0051L-0x49d630L-0x49a923L-0x497ef6L-0x49
52a8L-0x49287bL-0x4927b6L-0x4926fbL-0x48f896L-0x48d149L-0x48d084L-0x48cfcdL-0x48cf16L-0x48ce5bL-0x48a4efL-0x4872
e9L-0x484ba4L-0x481e93L-0x481dceL-0x481d17L-0x47f2f8L-0x47c75dL-0x47c698L-0x47c5e1L-0x47c52aL-0x47c473L-0x47c3bc
L-0x47c305L-0x47c24eL-0x47c197L-0x479a54L-0x476b2aL-0x476a69L-0x4769aeL-0x4743d9L-0x471120L-0x47105bL-0x470fa4L-
0x47430aL-0x4768edL-0x476844L-0x47678dL-0x4766d6L-0x47661fL-0x47426bL-0x470e40L-0x470d7bL-0x470cc4L-0x470c0dL-0x
470b52L-0x46eb2dL-0x46b588L-0x4693f7L-0x465fc8L-0x46608dL-0x466140L-0x46393aL-0x460a14L-0x45e2bfL-0x45b168L-0x45
8cebL-0x455a2eL-0x4532e9L-0x45067fL-0x4505beL-0x450507L-0x45321aL-0x455963L-0x458c1cL-0x45b09dL-0x45aff4L-0x45af
3dL-0x45e139L-0x46088aL-0x46386bL-0x465efdL-0x465e54L-0x465d9dL-0x469271L-0x46b402L-0x46e9a7L-0x4709daL-0x4740e5
L-0x476282L-0x4761d9L-0x47611eL-0x474046L-0x47093bL-0x46e908L-0x46b363L-0x4691d2L-0x465cf0L-0x4637ccL-0x46067dL-
0x460742L-0x4607f9L-0x45e09aL-0x45ae90L-0x458b7dL-0x455756L-0x45581bL-0x4558d2L-0x45317bL-0x45045eL-0x450399L-0x
```

```
4502deL-0x44df1eL-0x44aa44L-0x44a97fL-0x44a8c8L-0x44a811L-0x44a75eL-0x44a6a7L-0x44a5f0L-0x44a539L-0x44a482L-0x44
a3cbL-0x44a310L-0x448524L-0x444e1dL-0x442b1aL-0x43f575L-0x43f63aL-0x43f6f1L-0x442bb9L-0x444ebcL-0x444f99L-0x4450
50L-0x442c88L-0x43f79aL-0x43d609L-0x439d9cL-0x439e61L-0x439f18L-0x437e24L-0x434a01L-0x4324d5L-0x42f09eL-0x42d122
L-0x429e4dL-0x429f12L-0x429fc9L-0x42a080L-0x42a137L-0x427a87L-0x42487dL-0x4247b8L-0x424701L-0x42464aL-0x42458fL-
0x4224c3L-0x41edc0L-0x41c953L-0x4195dbL-0x4175b4L-0x413ea5L-0x411dbfL-0x40e5f9L-0x40c8a6L-0x408f8eL-0x408ec9L-0x
408e0eL-0x406c87L-0x402b9aL-0x402ad9L-0x402a26L-0x406bb8L-0x408d4dL-0x40c7d7L-0x40e3bcL-0x40e499L-0x40e550L-0x41
1cf0L-0x413ddaL-0x413d31L-0x413c7aL-0x413bc3L-0x413b0cL-0x4174e5L-0x419230L-0x41930dL-0x4193c4L-0x41947bL-0x4195
32L-0x41c884L-0x41ecf1L-0x41ec48L-0x41eb91L-0x41eadaL-0x41ea27L-0x41e970L-0x41e8b9L-0x42233dL-0x42413fL-0x424096
L-0x423fdfL-0x423f28L-0x423e71L-0x42229eL-0x41e755L-0x41e690L-0x41e5d9L-0x4221cfL-0x423db0L-0x427793L-0x42970fL-
0x429666L-0x4295abL-0x4276f4L-0x423d11L-0x422130L-0x41e530L-0x41e46bL-0x41e3b4L-0x41c72eL-0x418bd9L-0x418c9eL-0x
418d55L-0x417221L-0x4136ccL-0x411a2cL-0x40decfL-0x40c45cL-0x4086faL-0x4087bfL-0x408876L-0x40c4fbL-0x40df6eL-0x41
1acbL-0x41376bL-0x413844L-0x4138fbL-0x417377L-0x418ea1L-0x418f7eL-0x419031L-0x4190e8L-0x41919fL-0x417446L-0x413a
5fL-0x411c51L-0x40e1afL-0x40e274L-0x40e32bL-0x40c738L-0x408cb2L-0x408bedL-0x408b32L-0x406b19L-0x40280bL-0x402746
L-0x40268fL-0x4025d8L-0x402525L-0x40246eL-0x4023b7L-0x402300L-0x402249L-0x402192L-0x4020dbL-0x402024L-0x401f6dL-
0x401eb6L-0x401e03L-0x401d4cL-0x401c99L-0x401be2L-0x401b2bL-0x401a74L-0x4019bdL-0x406825L-0x408073L-0x40814cL-0x
408203L-0x40c2d6L-0x40d9b6L-0x40da93L-0x40db4aL-0x411738L-0x4133d4L-0x416fe4L-0x4188e5L-0x41883cL-0x418785L-0x41
86ceL-0x418617L-0x418560L-0x4184a9L-0x416f45L-0x41310cL-0x4115e2L-0x40d860L-0x40c180L-0x407fd4L-0x406786L-0x4019
10L'

'''
    @:param must be the switch jump address
    :return wasd's goto address by dic
'''
def get_switch_table(jump_ea):
    res = {}

    si = idaapi.get_switch_info_ex(jump_ea)
    assert si!=None
    results = idaapi.calc_switch_cases(jump_ea, si)
    wasd = [87,65,83,68]
    for idx in xrange(len(results.cases)):
        # muti case constrain will cause one case process
        cur_case = results.cases[idx]
        constrain = False
        for cidx in xrange(len(cur_case)):
            if cur_case[cidx] in wasd:
                # print "case: %d" % cur_case[cidx]
                res[results.targets[idx]] = cur_case[cidx]
                constrain = True
        if constrain==True:
            # print "  goto 0x%x" % results.targets[idx]
            pass
    return res

'''
    @:param must be the function address
    :return switch jump address
'''
def get_sjump_addr(caller_ea):
    func = idaapi.get_func(caller_ea)
    start_ea = func.startEA
    end_ea   = func.endEA

    pattern = 'FF E0' # push eax
    addr = start_ea
    addr = idc.FindBinary(addr, SEARCH_DOWN | SEARCH_NEXT, pattern)
    assert addr != idc.BADADDR and addr<=end_ea
    # print hex(addr),idc.GetDisasm(addr)
    return addr
```

```
'''
    by call locate_sjump_addr && get_switch_table
    to get the value of case belong

    @:param must be the refs address
    :return value
'''
def get_case_value(caller_ea=0x048D0B1, refs_ea=0x048D12F):
    jump_addr  = get_sjump_addr(caller_ea)
    wasd_table = get_switch_table(jump_addr)

    # from refs_ea to up untill first match wasd_table_re
    min_addr   = 0x1000
    target_addr= 0
    for addr in wasd_table:
        if refs_ea >= addr and (refs_ea-addr) < min_addr:
            min_addr = refs_ea-addr
            target_addr = addr
        # if caller_ea == 0x4184d6:
        #     print(hex(addr), wasd_table[addr])
        #     print('target_addr:', hex(target_addr), 'refs_ea', hex(refs_ea))

    return chr(wasd_table[target_addr])

'''
    format the addr from str to int
'''
def fm(addr_in_str):
    res = addr_in_str[2:]
    if res[-1]=='L':
        res = res[:-1]
    return int(res, 16)

get_case_value()

print '\n'*0x10000
print '[+] transvering datas'
caller_ls      =  s.split('-')
caller_refs_ls = s2.split('-')
res = ''
for i in range(len(caller_ls)):
    per = get_case_value(fm(caller_ls[i]), fm(caller_refs_ls[i]))
    res+=per
    print '[+] get %c from'%per, caller_ls[i], caller_refs_ls[i], len(caller_ls)-i-1+1
res = 'S'+res[::-1]
print '[+] path: '+res

import hashlib
print '[+] flag{'+hashlib.md5(res).hexdigest()+'}'

'''
print:
........
[+] path: SSSSSSSSSDDDDDDWWWWAAWWAAWWDDDDDDDDDDDDDDDDDDDDSSDDSSAASSSSAAAAWWAAWWWWAASSSSSSAASSDDSSSSDDWWWWDDSSDDD
DWWDDDDDDWWAAAAWWDDDDWWAAWWWWDDSSDDSSSSSSSSSSDDDDSSAAAASSSSSSAASSSSAAWWAASSSSDDDDDDDDDDSSDDSSAASSSSAASSSSSSSSDDW
WWWWWDDWWWWDDWWWWDDSSSSSSSSAASSSSDDDDSSDDDDWWDDSSDDSSDDDDDDDDSSDDSSSSDDDDSSDDSSSSSSDDSSSSDDDDSSSSDDDDDDSSSSDDSSD
SSASSSSAASSDDSSAASSDDDDDDSSDDDDWWDDSSSSSSDDDDWWAAWWWWDDDDSSSSDDDDDDSSAASSSSSSDDDDDDDDSSDDDDSSSSSSDDWWDDDDDDSSSSS
SSSAASSDDSSSSSSAASSDDS
[+] flag{078c8fbc1d0d033f663dcc58e899c101}
```

```
'''
```


Crypto(3题)
PWN(4题)
Reverse(3题)
61支队伍攻克
125pt
题目名称：baby_maze
题目类型：Reverse
38支队伍攻克
176pt
题目名
答案正确
题目类型：Reverse