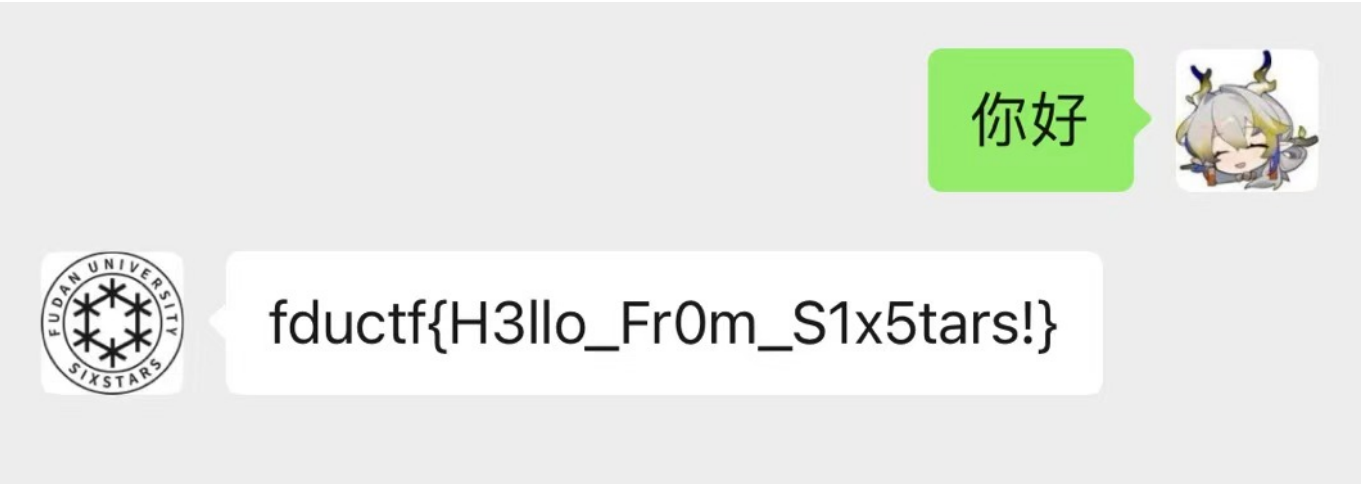
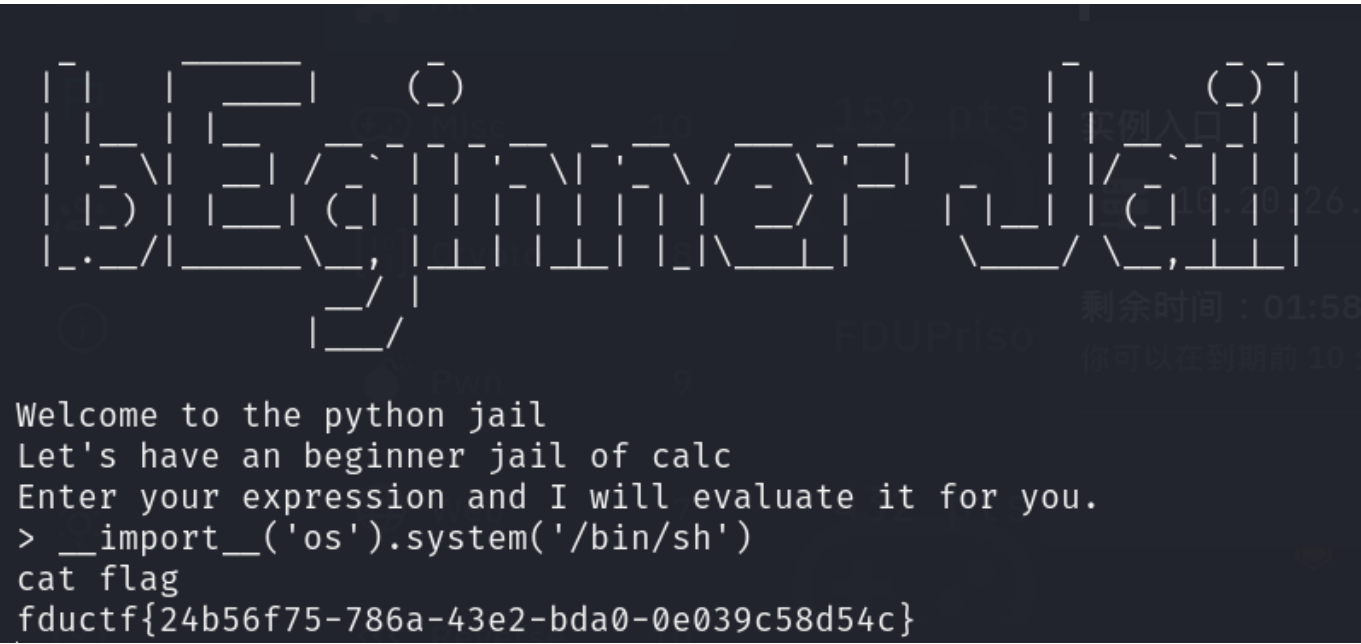


Misc

签到



FDUKindergarten

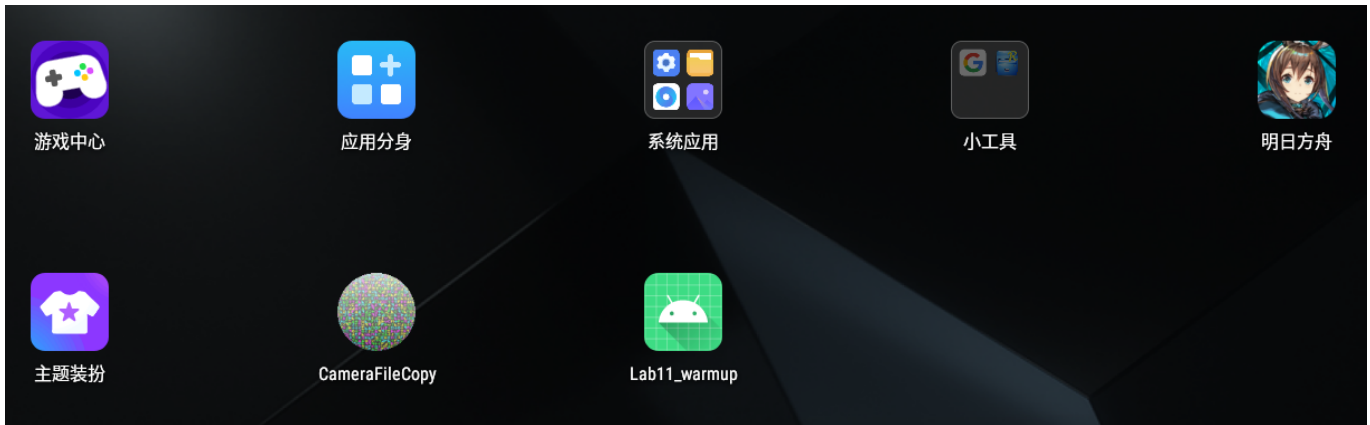


问卷反馈

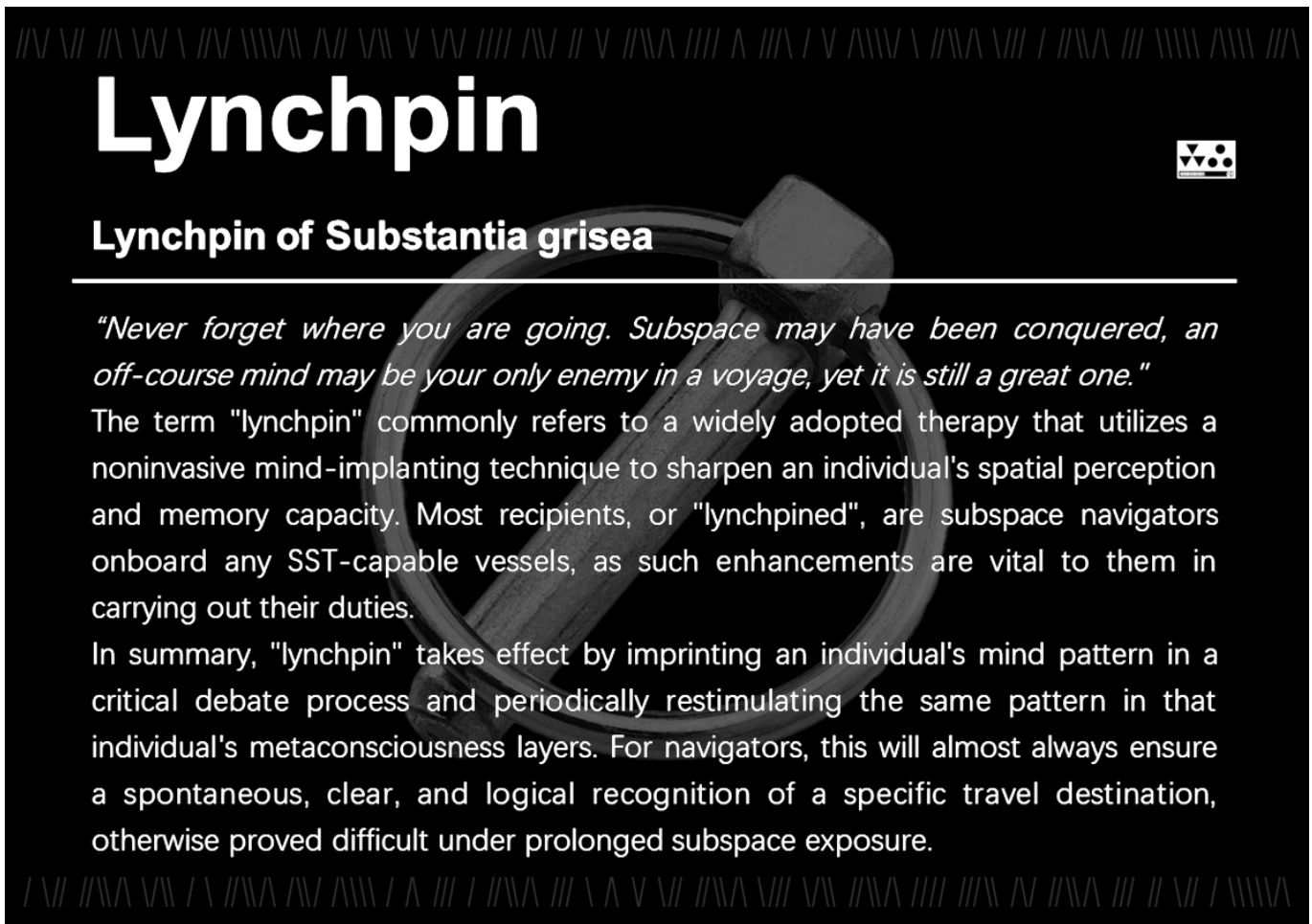
flag 在最后一个问题后

* 11. 请问您还有其他想要补充的意见或建议吗？(fductf{we11_9e_9ette3})

什么是Cimbar码



在模拟器中利用 CameraFileCopy 扫描 Essay.gif 得到 文件，查看 16 进制可知为 png 文件，重命名后打开



Lynchpin ? 好熟悉的单词呢 ()

注意到图片最上方和最下方部分，把 / 当做 .，\ 当做 -，对照摩斯电码，得到 flag 为

`fductf{lynchpin_haven't_be_s0lved_yet_please_stand_by_h3r_side}` 出题人是舟批 (确信)

Crypto

Alice 与 Bob 的小纸条

根据字母出现的频率建立映射 { 'c': 'e', 'C': 'E', 'w': 't', 'W': 'T', 'b': 'a', 'B': 'A', 'q': 'o', 'Q': 'O', 'u': 's', 'U': 'S', 'x': 'n', 'X': 'N', 't': 'i', 'T': 'I', 'i': 'r', 'I': 'R', 'l': 'h', 'L': 'H', 'm': 'd', 'M': 'D', 'd': 'c', 'D': 'C', 'e': 'l',

'E': 'L', 'z': 'u', 'Z': 'U', 'v': 'g', 'V': 'G', 'r': 'f', 'R': 'F', 's': 'y', 'S': 'Y', 'f': 'w', 'F': 'W', 'j': 'p', 'J': 'P', 'n': 'm', 'N': 'M', 'k': 'b', 'K': 'B', 'y': 'v', 'Y': 'V', 'o': 'k', 'O': 'K', 'h': 'q', 'H': 'Q', 'g': 'j', 'G': 'J', 'a': 'x', 'A': 'X', 'p': 'z', 'P': 'Z'} 得到解密后的文本 · 其中：The flag begins with fductf. Then comes the left brace. Contents inside two braces are WrodFerqency, which indicates you should analyze word frequency to solve this problem. And dont forget the right brace. That is, fductf, left brace, WrodFerqency, and right brace. 可得 flag 为 fductf{WrodFerqency}

Jeff Dean 笑话

```
from Crypto.Util.number import long_to_bytes

# 公钥(public key)
e = 65537
n =
7526068147102161455028249098110953254006476818951996300378885344962814823849109458
3307996632151408412567215312977037223220187737015088671579053426702046623459412581
50244652352208534643038915031081173693366320086362291

# 私钥(private key)
p = 8193423899118349
q =
9185498321296426312173572093061567893427873612682060939417964833837894845427087985
9729501096693466390415006413417247098810255220905151391173204390926205228934707696
6590217809538667560583127015378375359

phi = (p - 1) * (q - 1)

for k in range(2, phi):
    if (k * phi + 1) % e == 0:
        d = (k * phi + 1) // e
        break

# 密文(cipher text)
cipher_text =
2568646804735741487243301229828507981070833525234799601925268878720580071056004739
4911536917837531203511886398091659599490709404149096155398508032206535805490958290
69981847238258190205803383184077318785393455355825137

plain_text = pow(cipher_text, d, n)

print(long_to_bytes(plain_text))
```

得到 flag 为 ``

eazy_sage.py

最后的凯撒加密才是重点罢 ()

```
def caesar_cipher(text, shift):
    result = ""
    for char in text:
        if char.isalpha():
            shift_amount = shift % 26
            if char.islower():
                result += chr((ord(char) - ord('a') + shift_amount) % 26 +
ord('a'))
            else:
                result += chr((ord(char) - ord('A') + shift_amount) % 26 +
ord('A'))
            else:
                result += char
    return result

for i in range(26):
    print(f'fdctf{{{caesar_cipher("ym1x1xhwd9y0", i)}}}')

```

注意到结果中有 `fdctf{th1s1scry9t0}`

Pwn

test-your-nc

```
$ nc 10.20.26.32 32817
fdctf{Iet_U5_57aRT_PwN_610c54341f11}

```

Web

草率的毕业设计

username 为 `YWRtaW4=`，鉴定为base64，解码得到admin 密码以 `fdctf` 开头，由此穷举得到盐为 `95qW`，再穷举得到 flag `fdctf{salt_is_too_short_40Ecc8BC06e0Fb8f}`

Reverse

baby64

根据名字猜测与 base64 有关，但是怎么试都不对，查看汇编发现一个的函数 `_Z3sssv`，它对 `ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/` 进行处理，交换相邻两个字符的位置，得到 `BADC FEHG JILKNMPORQTSVUXWZYbadcf ehg jilknmporqtsvuxwzy1032547698/+`，以此解密即可得到 flag `fdctf{M4in_1s_n0t_the_fir3t_0ne}`

easy_python

```
import dis
import marshal
pyc_file_path = "./easy_python"

```

```
with open(pyc_file_path, "rb") as f:
    f.read(16) # 跳过头部
    code_object = marshal.load(f)

dis.dis(code_object)
```

反汇编得到字节码 · 利用 AI 编写解密代码

```
import binascii

def mm(key):
    N = 256
    s = list(range(N))
    j = 0
    key_length = len(key)

    for i in range(N):
        j = (j + s[i] + key[i % key_length]) % N
        s[i], s[j] = s[j], s[i]

    return s

def nn(s, data):
    i = 0
    j = 0
    N = 256
    result = []

    for byte in data:
        i = (i + 1) % N
        j = (j + s[i]) % N
        s[i], s[j] = s[j], s[i]
        k = s[(s[i] + s[j]) % N]
        result.append(byte ^ k)

    return bytes(result)

# 密钥
key = b'FDUCTF{2024}'

# 已知的加密数据 (check_data)
check_data =
binascii.unhexlify('f9ecf8f97b8f96baecc607004ec26724623cee86ece84a4581f8c063')

# 初始化密钥流
s = mm(key)

# 解密
decrypted_flag = nn(s, check_data)
```

```
# 打印解密后的 flag
print(decrypted_flag.decode())
```

flag 为 `fductf{Y0ur4r3g00d@tPyth0n!}`

CSharpReverse

使用 dnSpy 分析 flag.dll 得到

```
public bool CheckText()
{
    string text = "hd3Q2fyvrbEfAr_}r{utcsc_!d4qeseAp";
    Random random = new Random(33554432);
    text = new string((from c in text
        orderby random.Next()
        select c).ToArray<char>());
    return this.textBox1.Text == text;
}
```

从而编写代码

```
class Program
{
    static void Main()
    {
        string text = "hd3Q2fyvrbEfAr_}r{utcsc_!d4qeseAp";
        Random random = new Random(33554432);
        text = new string((from c in text
            orderby random.Next()
            select c).ToArray());
        Console.WriteLine(text);
    }
}
```

得到 flag 为 `fductf{c_shArp_revErse!A23qQdyb4}`

ez_android

使用 jadx · 根据 AndroidManifest.xml 找到 `com.example.lab11_warmup.MainActivity`

```
/* Loaded from: classes.dex */
public class MainActivity extends AppCompatActivity {
    public native boolean Check(String str);

    static {
        System.loadLibrary("lab11_warmup");
    }

    /* JADX INFO: Access modifiers changed from: protected */
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, android
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        setContentView(R.layout.activity_main);
        final TextView textView = (TextView) findViewById(R.id.flag);
        ((Button) findViewById(R.id.button)).setOnClickListener(new View.OnClickListener() { // from c
            @Override // android.view.View.OnClickListener
            public void onClick(View view) {
                if (MainActivity.this.Check(textView.getText().toString())) {
                    Toast.makeText(MainActivity.this.getApplicationContext(), "RIGHT!", 0).show();
                } else {
                    Toast.makeText(MainActivity.this.getApplicationContext(), "WRONG!", 0).show();
                }
            }
        });
    }
}
```

发现 check 方法是 native 的，保存反编译资源，使用 ida 处理 .so 文件

```

if ( !(unsigned __int8)aaa(v3) )
    return 0;
v7 = 0LL;
*(_OWORD *)dest = 0LL;
strncpy(dest, v3 + 5, 0xBu);
s1[0] = dest[0] ^ 5;
s1[1] = dest[1] ^ 5;
s1[2] = dest[2] ^ 5;
s1[3] = dest[3] ^ 5;
s1[4] = dest[4] ^ 5;
s1[5] = dest[5] ^ 5;
s1[6] = dest[6] ^ 5;
s1[7] = dest[7] ^ 5;
s1[8] = dest[8] ^ 5;
s1[9] = dest[9] ^ 5;
s1[10] = dest[10] ^ 5;
s1[11] = 0;
return strcmp(s1, "p0r3_1s_fUn") == 0;

```

```

bool __cdecl aaa(const char *a1)
{
    return *a1 == 70
        && a1[1] == 76
        && a1[2] == 65
        && a1[3] == 71
        && a1[4] == 123
        && a1[16] == 125
        && (unsigned __int8)(a1[5] - 123) >= 0xB5u
        && (unsigned __int8)(a1[6] - 123) >= 0xB5u
        && (unsigned __int8)(a1[7] - 123) >= 0xB5u
        && (unsigned __int8)(a1[8] - 123) >= 0xB5u
        && (unsigned __int8)(a1[9] - 123) >= 0xB5u
        && (unsigned __int8)(a1[10] - 123) >= 0xB5u
        && (unsigned __int8)(a1[11] - 123) >= 0xB5u
        && (unsigned __int8)(a1[12] - 123) >= 0xB5u
        && (unsigned __int8)(a1[13] - 123) >= 0xB5u
        && (unsigned __int8)(a1[14] - 123) >= 0xB5u
        && (unsigned __int8)(a1[15] - 48) < 0x4Bu;
}


```

由 aaa 函数确定 flag 前5位对应的 ascii 码为70 76 65 71 123，最后一位为 125，即FLAG{...}，再将 p0r3_1s_fUn 中每个字符分别与 5 异或得到 u5w6Z4vZcPk，flag 即为 FLAG{u5w6Z4vZcPk}

FLAG{u5w6Z4vZcPk}|

check

This is warmup challenge

 RIGHT!

functions

用 z3 求解

```
from z3 import *

solver = Solver()

v0 = BitVec('v0', 64)
v1 = BitVec('v1', 64)
v2 = BitVec('v2', 64)
v3 = BitVec('v3', 64)

solver.add(((v3 ^ (~v3 + v1) | v1) & v0) & v2) == 0x2050472E53002A03)
solver.add((v1 & (v1 & v2 | ~(v1 | v0)) & v0 | v3 & v0 & v2) ==
0x4860466062306422)
solver.add(((v3 & ~(v2 & v0) | (v0 + v2) & v2 | (v1 + v3) & v1 & ~v0) ^ v1) ==
0x2A00033A32352E61)
solver.add(((v2 - v3) ^ (v0 - v1)) == 0x1146CDC7BFA3E00E)
solver.add((v2 + v0 - v3 + v1) * (v1 + v0 + v3 - v2) == 0x30820AD98D807A4)
solver.add((v2 - v1 - v0 - v3) % 114514 == 75028)
solver.add((v0 * v1 * v2 * v3) % 1919810 == 567916)

model = solver.model()
print(f"v0: {model[v0]}, v1: {model[v1]}, v2: {model[v2]}, v3: {model[v3]}")
```

解出 v0~v3 · 再解码为字符串 `fductf{Y0u_h@ve_kn0wn_th3_z2z!!}`