Stack Guard Based on LLVM Pass

李有霖 202021080918

1. Background - LLVM system

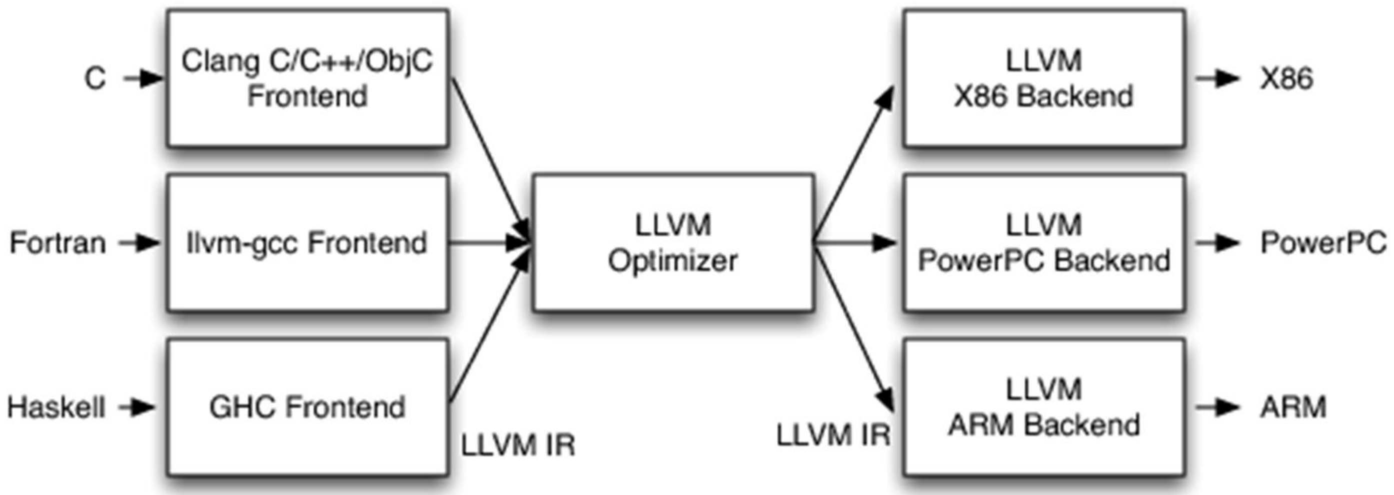


Figure 1. Overview of LLVM system

1. Background - LLVM pass

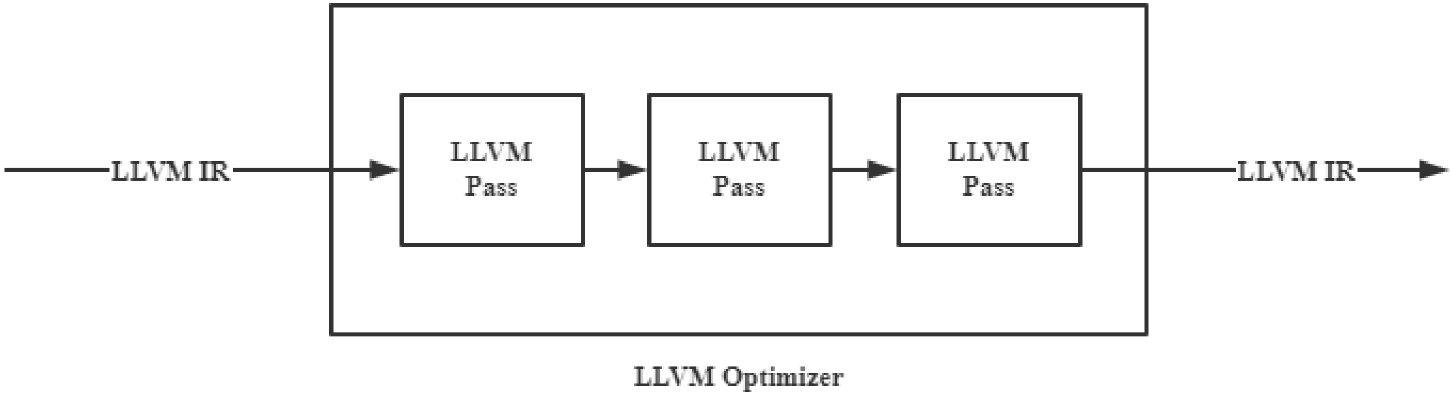


Figure 2. Internal of LLVM Optimizer

2. Overview

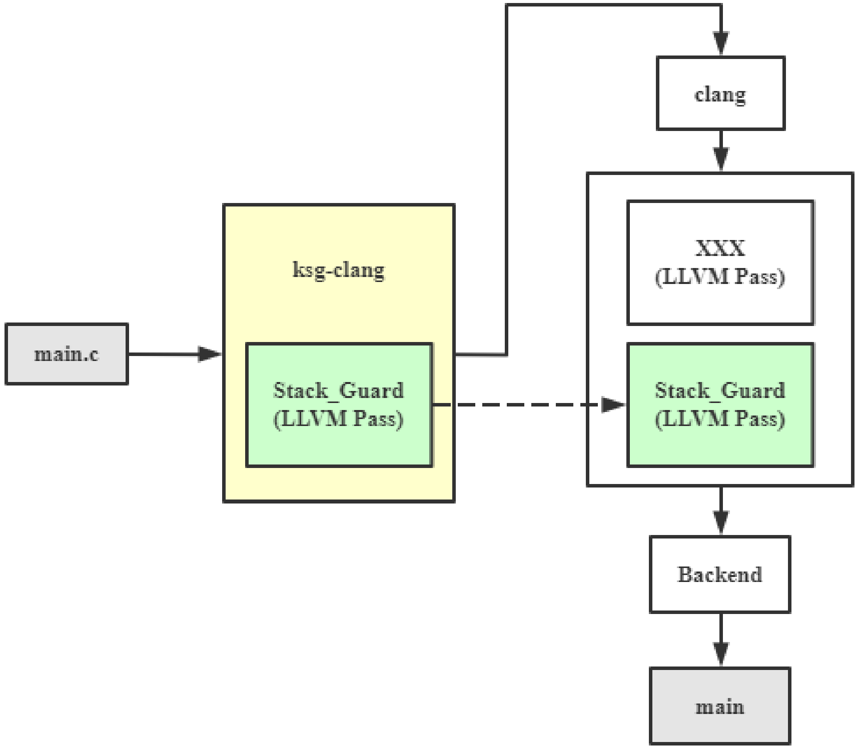


Figure 3. Overview of k-s-g

3. Methodology

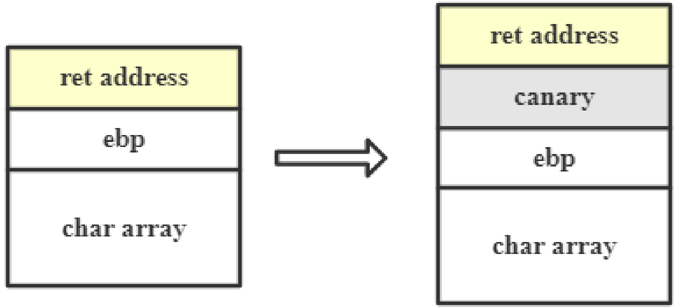


Figure 4. Modification on stack frame

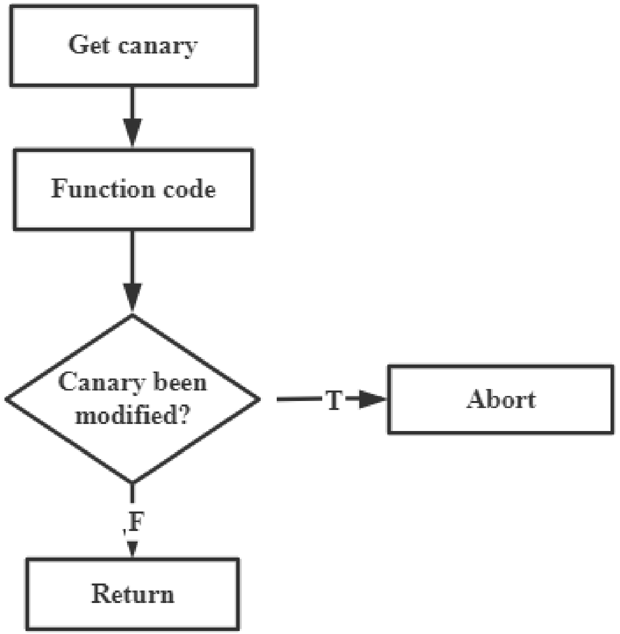
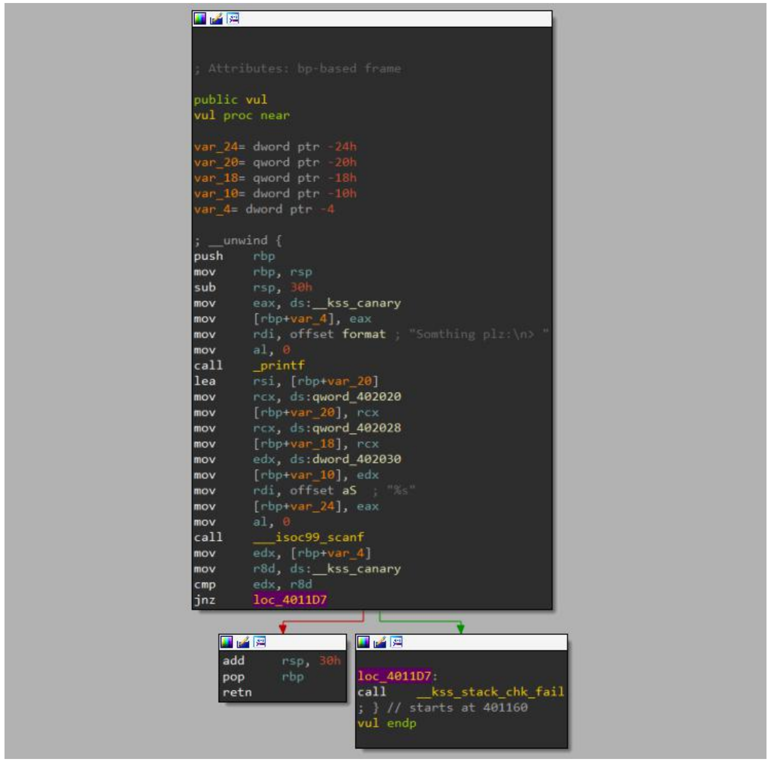


Figure 5. Modification on control flow

4. Implementation

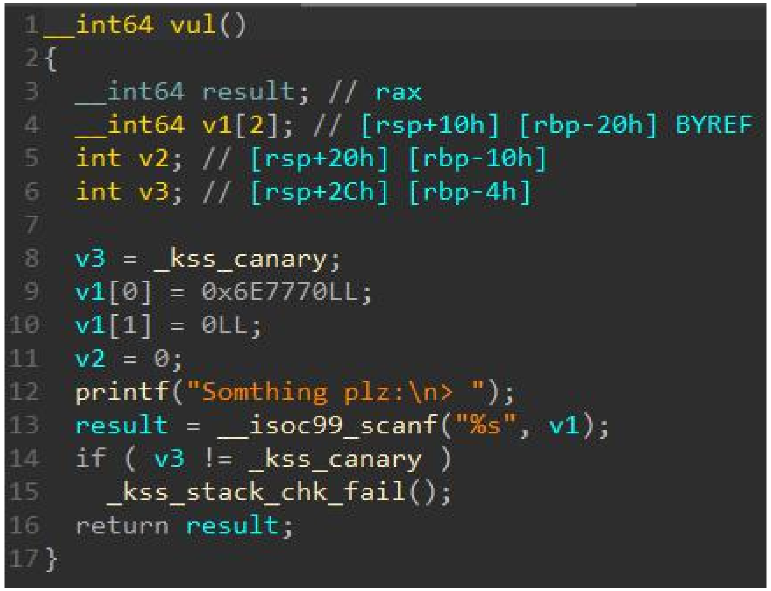


```
; Attributes: bp-based frame

public vul
vul proc near

var_24= dword ptr -24h
var_20= qword ptr -20h
var_18= qword ptr -18h
var_10= dword ptr -10h
var_4= dword ptr -4

; __unwind {
push    rbp
mov     rbp, rsp
sub     rsp, 30h
mov     eax, ds:__kss_canary
mov     [rbp+var_4], eax
mov     rdi, offset format ; "Somthing plz:\n> "
mov     al, 0
call    _printf
lea     rsi, [rbp+var_20]
mov     rcx, ds:qword_402020
mov     [rbp+var_20], rcx
mov     rcx, ds:qword_402028
mov     [rbp+var_18], rcx
mov     edx, ds:dword_402030
mov     [rbp+var_10], edx
mov     rdi, offset aS  ; "%s"
mov     [rbp+var_24], eax
mov     al, 0
call    ___isoc99_scanf
mov     edx, [rbp+var_4]
mov     r8d, ds:__kss_canary
cmp     edx, r8d
jnz     loc_4011D7
```

```
add     rsp, 30h
pop     rbp
retn
```

```
loc_4011D7:
call    __kss_stack_chk_fail
; } // starts at 401160
vul endp
```

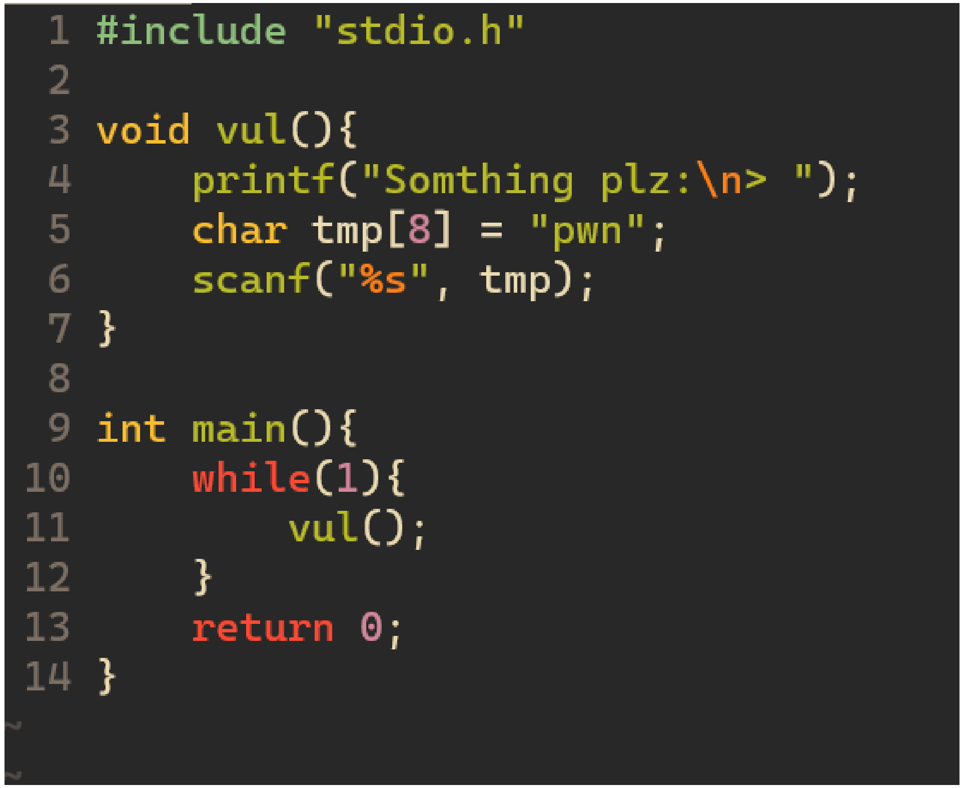
Figure 6. After implementation



```
__int64 vul()
{
  __int64 result; // rax
  __int64 v1[2]; // [rsp+10h] [rbp-20h] BYREF
  int v2; // [rsp+20h] [rbp-10h]
  int v3; // [rsp+2Ch] [rbp-4h]

  v3 = _kss_canary;
  v1[0] = 0x6E7770LL;
  v1[1] = 0LL;
  v2 = 0;
  printf("Somthing plz:\n> ");
  result = __isoc99_scanf("%s", v1);
  if ( v3 != _kss_canary )
    _kss_stack_chk_fail();
  return result;
}
```

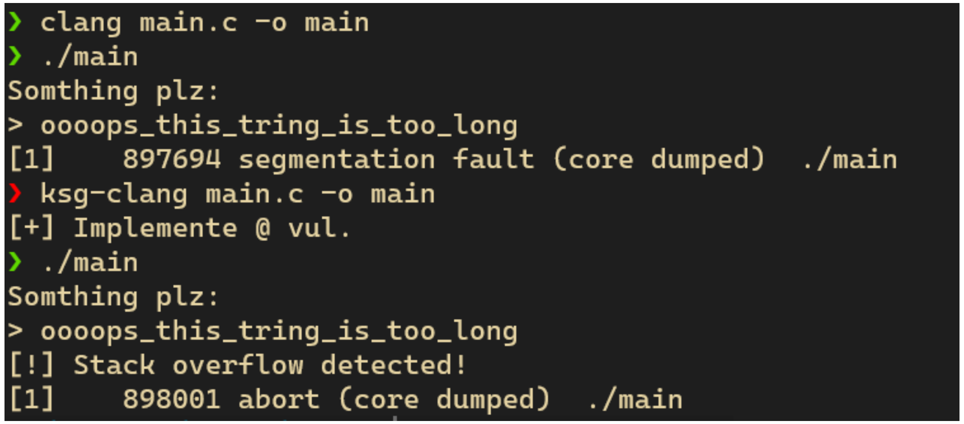
Figure 7. Pseudo-code

5. Experiment



```c
#include "stdio.h"

void vul(){
    printf("Somthing plz:\n> ");
    char tmp[8] = "pwn";
    scanf("%s", tmp);
}

int main(){
    while(1){
        vul();
    }
    return 0;
}
```

Figure 8. Vulnerable procedure

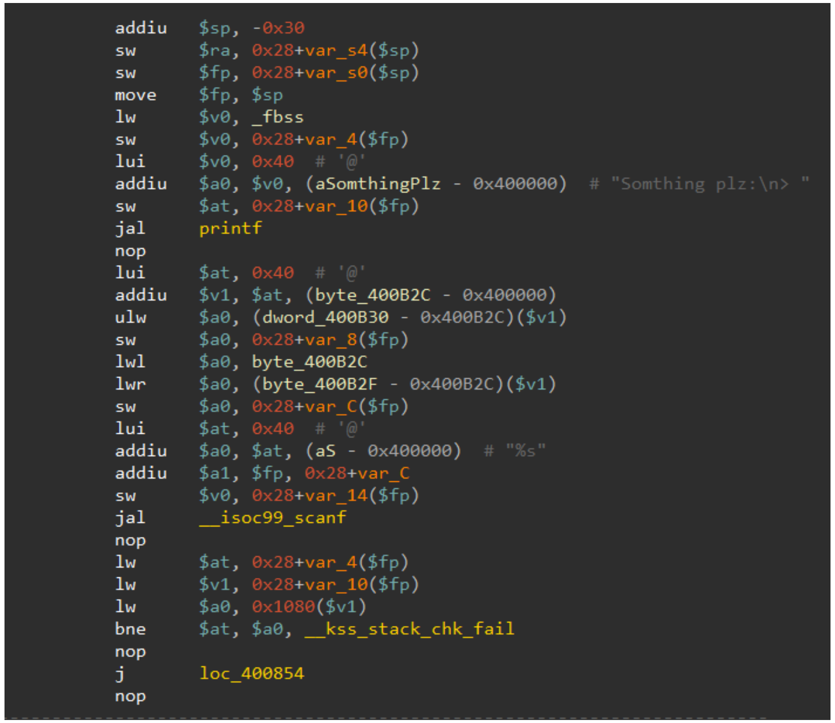


```
❯ clang main.c -o main
❯ ./main
Somthing plz:
> oooops_this_tring_is_too_long
[1]    897694 segmentation fault (core dumped)  ./main
❯ ksg-clang main.c -o main
[+] Implemente @ vul.
❯ ./main
Somthing plz:
> oooops_this_tring_is_too_long
[!] Stack overflow detected!
[1]    898001 abort (core dumped)  ./main
```

Figure 9. Experimental result

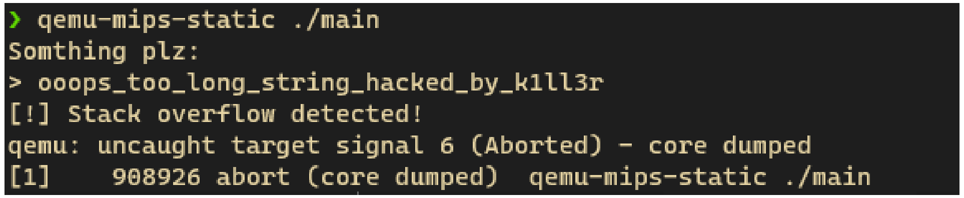
MIPS

+ Also works well on MIPS :)



```
addiu   $sp, -0x30
sw      $ra, 0x28+var_s4($sp)
sw      $fp, 0x28+var_s0($sp)
move    $fp, $sp
lw      $v0, _fbss
sw      $v0, 0x28+var_4($fp)
lui     $v0, 0x40  # '@'
addiu   $a0, $v0, (aSomthingPlz - 0x400000)  # "Somthing plz:\n> "
sw      $at, 0x28+var_10($fp)
jal     printf
nop
lui     $at, 0x40  # '@'
addiu   $v1, $at, (byte_400B2C - 0x400000)
ulw     $a0, (dword_400B30 - 0x400B2C)($v1)
sw      $a0, 0x28+var_8($fp)
lwl     $a0, byte_400B2C
lwr     $a0, (byte_400B2F - 0x400B2C)($v1)
sw      $a0, 0x28+var_C($fp)
lui     $at, 0x40  # '@'
addiu   $a0, $at, (aS - 0x400000)  # "%s"
addiu   $a1, $fp, 0x28+var_C
sw      $v0, 0x28+var_14($fp)
jal     __isoc99_scanf
nop
lw      $at, 0x28+var_4($fp)
lw      $v1, 0x28+var_10($fp)
lw      $a0, 0x1080($v1)
bne     $at, $a0, __kss_stack_chk_fail
nop
j       loc_400854
nop
```

Figure 10. Implementation for MIPS



```
❯ qemu-mips-static ./main
Somthing plz:
> ooops_too_long_string_hacked_by_k1ll3r
[!] Stack overflow detected!
qemu: uncaught target signal 6 (Aborted) - core dumped
[1]    908926 abort (core dumped)  qemu-mips-static ./main
```

Figure 11. Experimental result for MIPS

NO PWN NO FUN

- https://github.com/RLee063/Courses/tree/master/k-s-g
- https://llvm.org/docs/WritingAnLLVMPass.html
- https://github.com/llvm/llvm-project/blob/main/llvm/lib/CodeGen/StackProtector.cpp