作業十：追蹤 execve 系統呼叫

學習目標：

1. 了解作業系統如何解析執行檔案的格式

2. 了解作業系統在處理 execve 時，是否立即載入執行檔案？或者只是修改 task_struct 中的 mm_struct?

題目：

1. 撰寫程式碼稱之為 myls，在程式碼中使用 execve 系列的任何 libc 函數，載入新的執行檔案 (ls)。注意：不需要使用 fork。

2. 請問作業系統如何載入執行檔案？附上程式碼的截圖，並約略說明。

3. 請問作業系統是否立即載入執行檔案到記憶體中？附上程式碼的截圖並約略說明

- 作業繳交：

  - 上述 1~3 都是繳交 pdf。其中(1)必須附上你的程式碼的截圖。
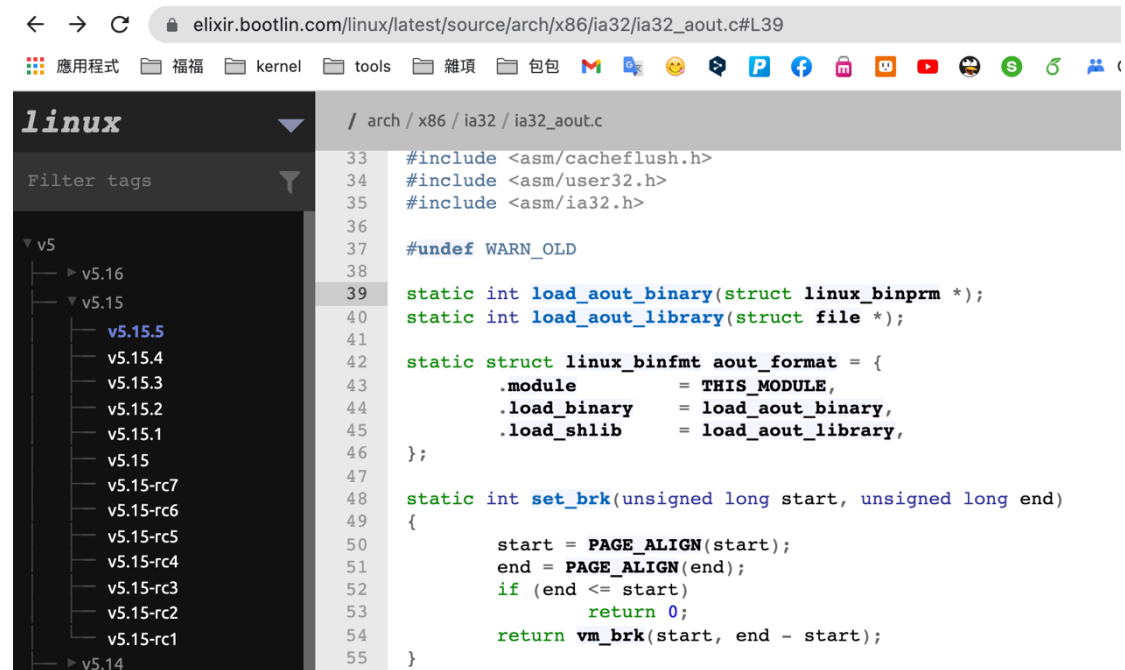
  - Pdf 中的學號、姓名（請隱藏個人資訊，例如：學號 687410007，姓名：羅 X 五)

繳交：

- 繳交期限：請參考網頁
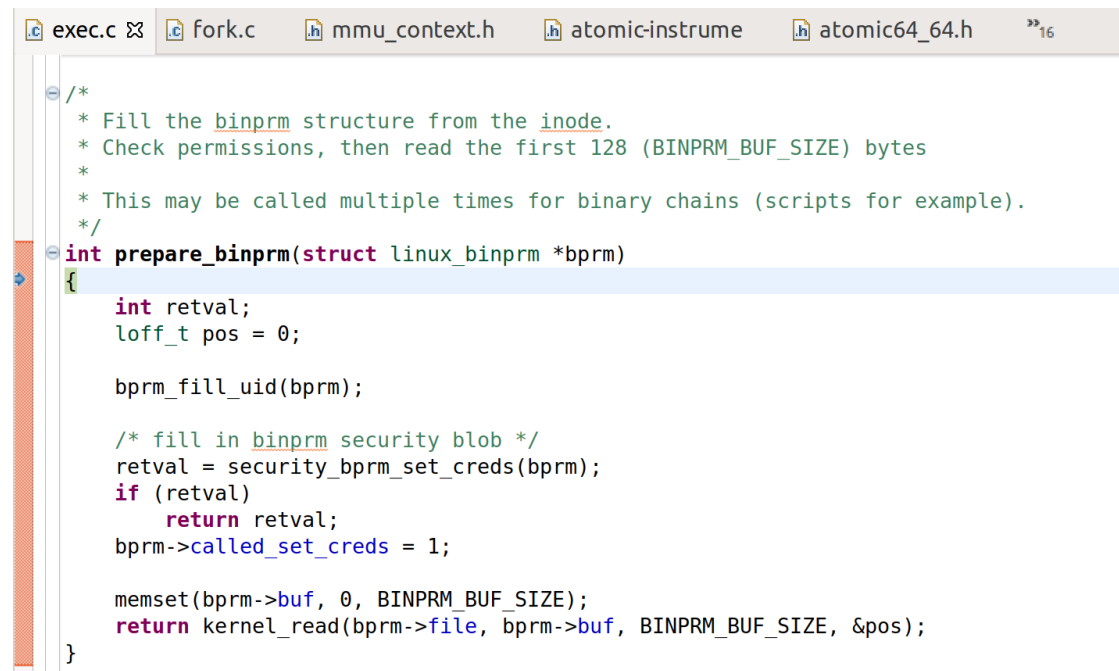
- 這次沒有 youtube 的說明。如果真的不會寫，記得去請教朋友。在你的報告上寫你請教了誰即可。

提示：

1. 可以將中斷點設定在 b do_execve

2. 圖如下

3. 圖如下

4. 圖如下

```
  exec.c    fork.c    mmu_context.h    read_write.c    module.c    »18

        /* This allows 4 levels of binfmt rewrites before failing hard. */
        if (bprm->recursion_depth > 5)
            return -ELOOP;

        retval = security_bprm_check(bprm);
        if (retval)
            return retval;

        retval = -ENOENT;
    retry:
        read_lock(&binfmt_lock);
        list_for_each_entry(fmt, &formats, lh) {
            if (!try_module_get(fmt->module))
                continue;
            read_unlock(&binfmt_lock);
            bprm->recursion_depth++;
            retval = fmt->load_binary(bprm);
            read_lock(&binfmt_lock);
            put_binfmt(fmt);
            bprm->recursion_depth--;
            if (retval < 0 && !bprm->mm) {
                /* we got to flush_old_exec() and failed after it */
                read_unlock(&binfmt_lock);
                force_sigsegv(SIGSEGV, current);
```
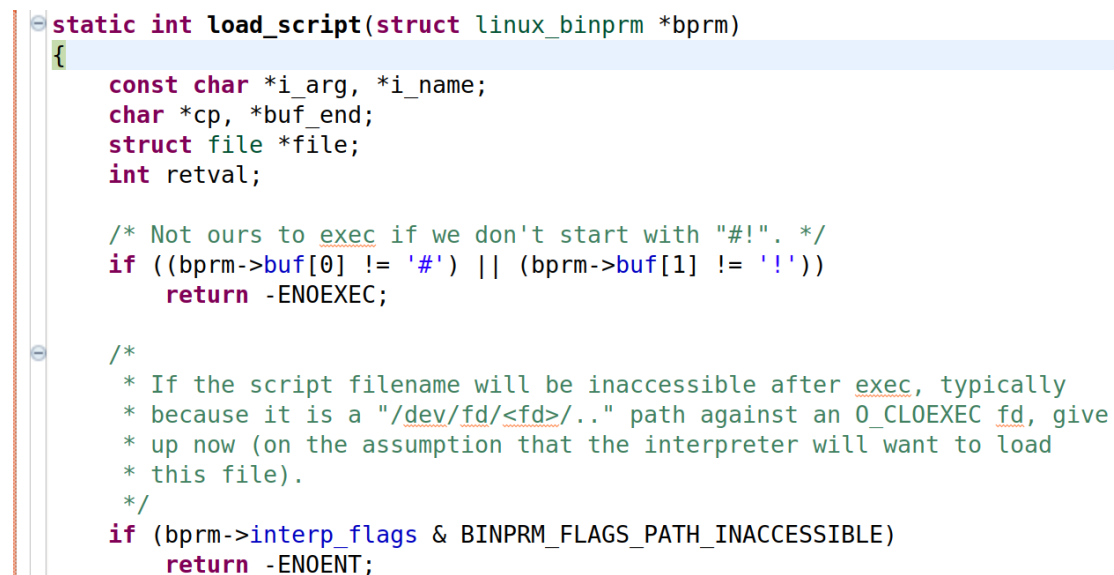
5. 圖如下

```
static int load_script(struct linux_binprm *bprm)
{
    const char *i_arg, *i_name;
    char *cp, *buf_end;
    struct file *file;
    int retval;

    /* Not ours to exec if we don't start with "#!". */
    if ((bprm->buf[0] != '#') || (bprm->buf[1] != '!'))
        return -ENOEXEC;

    /*
     * If the script filename will be inaccessible after exec, typically
     * because it is a "/dev/fd/<fd>/.." path against an O_CLOEXEC fd, give
     * up now (on the assumption that the interpreter will want to load
     * this file).
     */
    if (bprm->interp_flags & BINPRM_FLAGS_PATH_INACCESSIBLE)
        return -ENOENT;
```

6. 圖如下

```c
static int load_elf_binary(struct linux_binprm *bprm)
{
    struct file *interpreter = NULL; /* to shut gcc up */
    unsigned long load_addr = 0, load_bias = 0;
    int load_addr_set = 0;
    char * elf_interpreter = NULL;
    unsigned long error;
    struct elf_phdr *elf_ppnt, *elf_phdata, *interp_elf_phdata = NULL;
    unsigned long elf_bss, elf_brk;
    int bss_prot = 0;
    int retval, i;
    unsigned long elf_entry;
    unsigned long interp_load_addr = 0;
    unsigned long start_code, end_code, start_data, end_data;
    unsigned long reloc_func_desc __maybe_unused = 0;
    int executable_stack = EXSTACK_DEFAULT;
    struct pt_regs *regs = current_pt_regs();
    struct {
        struct elfhdr elf_ex;
```
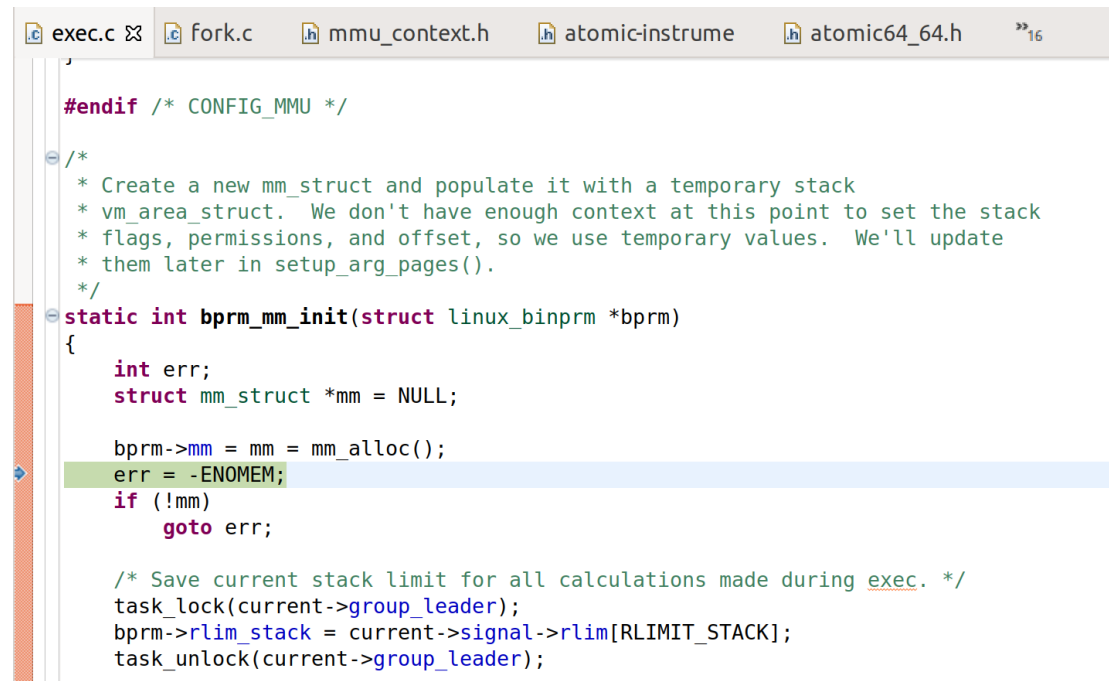
7. 圖如下

Debug ⊠   Project Explorer

- RunningLinuxKernel 5.0 [C/C++ Remote Application]
  - vmlinux [1]
    - Thread #1 1 (CPU#0 [running]) (Suspended : Step)
      - load_elf_binary() at binfmt_elf.c:691 0xffffffff81538ea4
      - search_binary_handler() at exec.c:1,656 0xffffffff81484b27
      - exec_binprm() at exec.c:1,698 0xffffffff81484d9d
      - __do_execve_file() at exec.c:1,818 0xffffffff814854b1
      - do_execveat_common() at exec.c:1,865 0xffffffff81485737
      - do_execve() at exec.c:1,882 0xffffffff81485838
      - __do_sys_execve() at exec.c:1,963 0xffffffff81485c94
      - __se_sys_execve() at exec.c:1,958 0xffffffff81485c4f
      - __x64_sys_execve() at exec.c:1,958 0xffffffff81485be9
      - do_syscall_64() at common.c:290 0xffffffff810075e6
      - <...more frames...>
  - gdb (9.2)

8. 圖如下


```c
#endif /* CONFIG_MMU */

/*
 * Create a new mm_struct and populate it with a temporary stack
 * vm_area_struct.  We don't have enough context at this point to set the stack
 * flags, permissions, and offset, so we use temporary values.  We'll update
 * them later in setup_arg_pages().
 */
static int bprm_mm_init(struct linux_binprm *bprm)
{
    int err;
    struct mm_struct *mm = NULL;

    bprm->mm = mm = mm_alloc();
    err = -ENOMEM;
    if (!mm)
        goto err;

    /* Save current stack limit for all calculations made during exec. */
    task_lock(current->group_leader);
    bprm->rlim_stack = current->signal->rlim[RLIMIT_STACK];
    task_unlock(current->group_leader);
```