



中國人民大學

RENMIN UNIVERSITY OF CHINA

信息学院

SCHOOL OF INFORMATION

软件安全分析
(二进制程序)

1. 二进制程序分析入门

授课教师：游伟 副教授

授课时间：周五14:00 – 16:30（教学三楼3505）

课程主页：<https://www.youwei.site/course/binary>

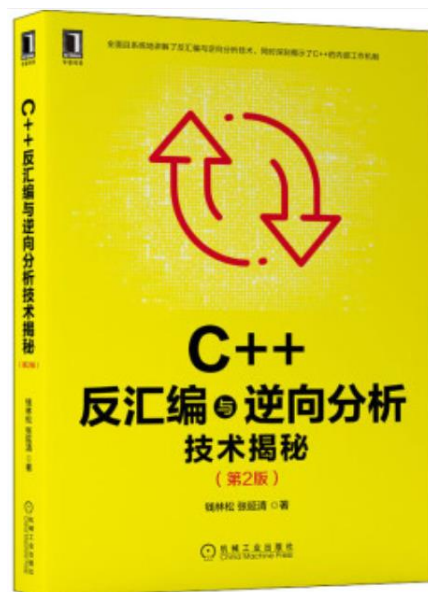
推荐图书



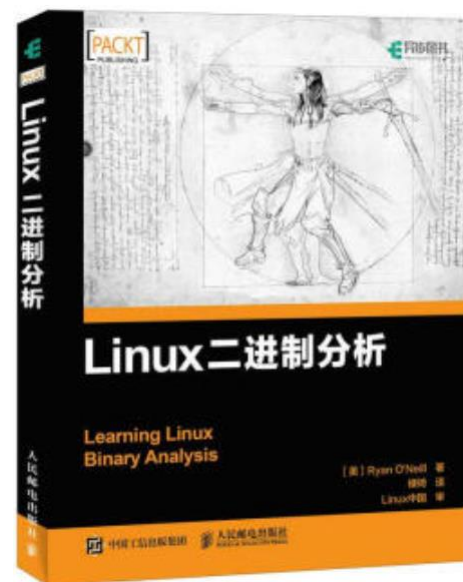
《逆向工程核心原理》
人民邮电出版社



《加密与解密》
电子工业出版社

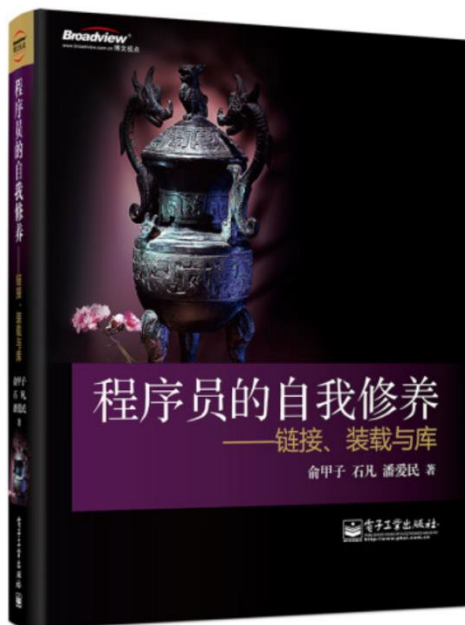


《C++反汇编与逆向分析
技术揭秘》
机械工业出版社

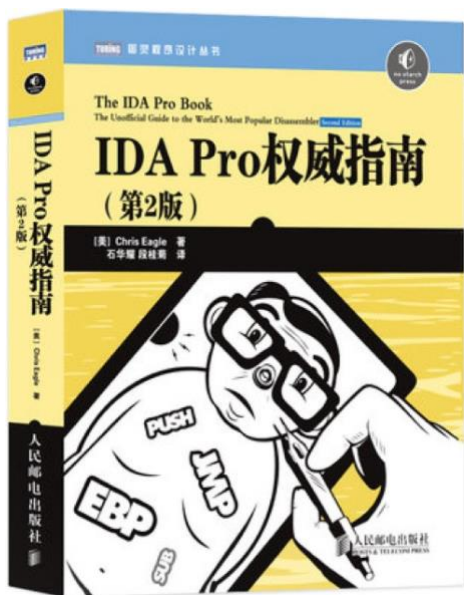


《Linux二进制分析》
人民邮电出版社

推荐图书



《程序员的自我修养——链接、装载与库》
电子工业出版社



《IDA Pro权威指南》
人民邮电出版社



《软件调试》
人民邮电出版社



《软件安全分析与应用》
清华大学出版社

在线培训课程



Windows逆向工程技术

讲师：吴春旺

<https://study.ichunqiu.com/student/course/57769>

引子：简易解压缩软件Decompress

有一个简易解压缩软件Decompress，具有如下功能和约束：

- 支持将指定的.tar或.rar压缩包文件中的内容解压缩到指定的文件路径中
- 注册用户无限制使用，非注册用户只能从每个压缩包中解压缩出至多3个文件
- 注册使用序列号保护机制，根据提供的邮箱和用户名计算出序列号
- 软件提供自动更新功能，可以替换部分组件

作为安全研究人员，你有以下几个待完成的任务：

- **软件破解**：破解软件的核心注册逻辑，达到无须注册也能无限制使用的效果
- **软件重构**：重构软件的代码与结构，达到添加额外功能或改变原有逻辑的效果
- **软件保护**：保护软件的代码与数据，达到反分析、反跟踪、反修改的效果

目录

1. 程序的结构与组成
2. 程序的生成与运行
3. 软件破解应用实例
4. 软件重构应用实例
5. 软件保护应用实例
6. 二进制程序分析技术概览

1 程序的结构与组成

- 工程文件 (CMakeLists): 指定软件项目编译构建流程
- 源代码文件 (.c/.cpp/.h): 由高级语言编写的程序源代码
- 目标文件 (.obj/.o): 由编译器编译生成的文件, 未进行外部符号重定位, 无法直接运行
- 静态库 (.lib/.a): 包含公用函数, 在程序编译时会被链接到目标代码中, 程序运行时将不再需要该静态库
- 动态库 (.dll/.so): 包含公用函数, 在程序编译时不会被链接到目标代码中, 在程序运行时才被载入内存并进行链接
- 可执行程序: 可以直接运行的程序文件

Decompress的构成

```
// 工程文件: CMakeLists.txt
1.  add_executable (Decompress main archive)
2.  add_library (main OBJECT "main.cpp" "main.h")
3.  add_library (archive OBJECT "archive.cpp" "archive.h")
4.  add_library (libRegister SHARED "register.cpp" "register.h")
5.  add_library (libTar "tar.cpp" "tar.h")
6.  add_library (libRar "rar.cpp" "rar.h")
7.  add_library (libProtect SHARED "protect.cpp" "protect.h")

8.  target_link_libraries (Decompress libRegister)
9.  target_link_libraries (Decompress libTar)
10. target_link_libraries (Decompress libRar)
```

 可执行文件

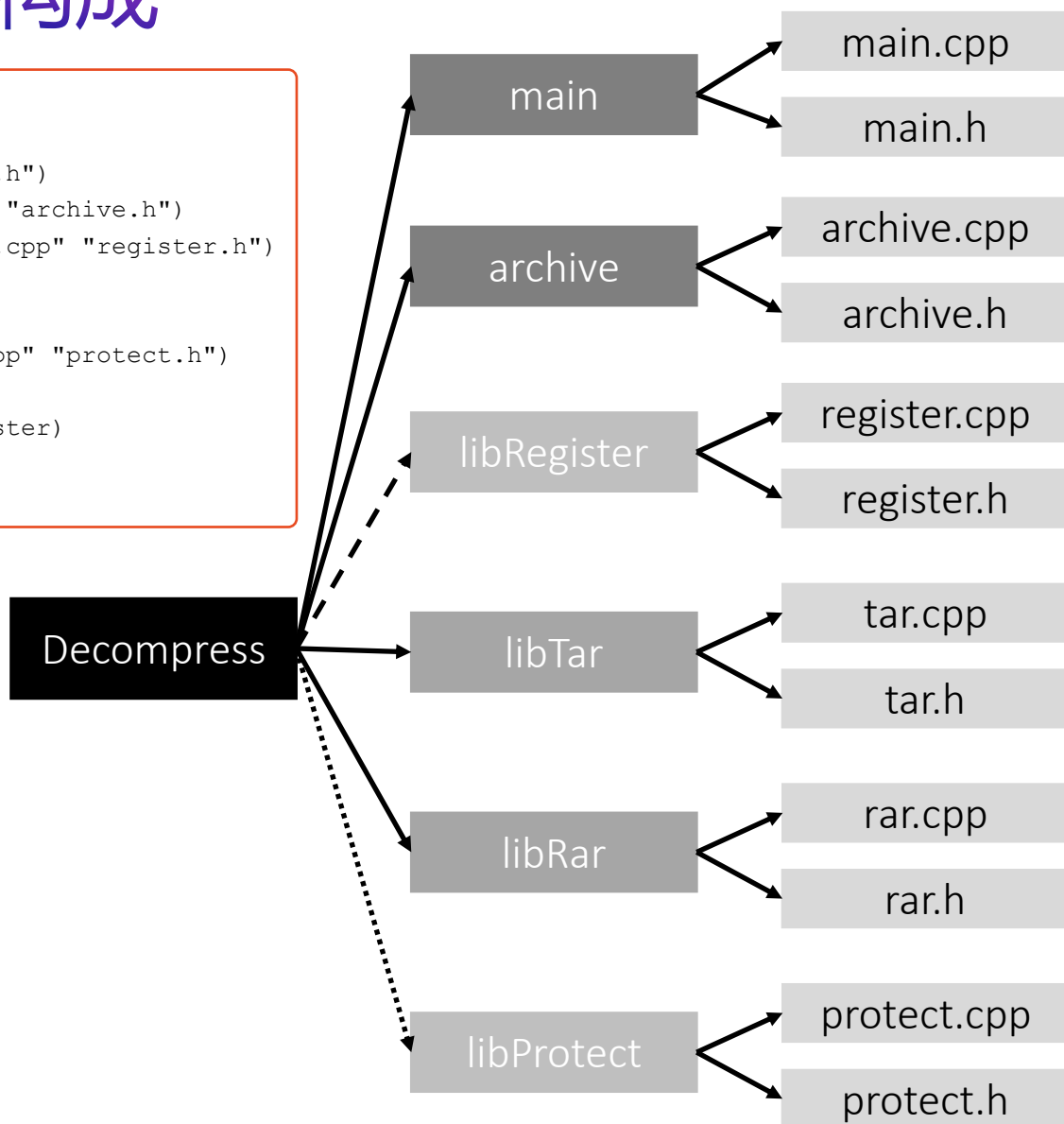
 目标文件

 静态链接库

 动态链接库

 源文件

 依赖关系
 依赖关系
 依赖关系



main

本模块中定义的符号，编译时确定定位。

其它模块中定义的符号，静态链接时确定定位。

其它模块中定义的符号，装载时动态链接确定定位。

其它模块中定义的符号，运行时动态链接确定定位。

```
1. extern "C" bool try_register(char *, char *, int);
2. extern "C" void do_extract(void);

3. int before_register(char *email, char *name) {
4.     int code;

5.     printf("Please input your email: ");
6.     scanf("%s", email);
7.     printf("Please input your name: ");
8.     scanf("%s", name);
9.     printf("Please input serial code: ");
10.    scanf("%d", &code);

11.    return code;
12.}

13.void before_extract(void) {
14.    printf("Please input the archive file path: ");
15.    scanf("%s", src);
16.    printf("Please input the destination path: ");
17.    scanf("%s", dest);
18.}

19.void do_update(void) {
20.    HMODULE dlib; FUNCPTR func;
21.    lib = LoadLibraryA("libProtect.dll");
22.    func = GetProcAddress(dlib, "try_protect");
23.    if (func) func();
24.    FreeLibrary(dlib);
25.}
```

```
26.extern "C" bool is_registered;
27.extern "C" char src[MAXLEN];
28.extern "C" char dest[MAXLEN];

29.int main(int argc, char **argv) {
30.    char email[MAXLEN], name[MAXLEN];
31.    int code;

32.    do_update();

33.    code = before_register(email, name);

34.    if (try_register(email, name, code))
35.    {
36.        is_registered = true;
37.        printf("Successfully registered!
38.                You can use full functionalities.\n");
39.    }
40.    else
41.    {
42.        is_registered = false;
43.        printf("Failed to register! You can extract
44.                at most 3 files from an archive.\n");
45.    }

46.    before_extract();
47.    do_extract();

48.    return 0;
49.}
```

archive

```
1. class Archive {
2. public:
3.     static bool is_tar(const char *);
4.     static bool is_rar(const char *);
5.     Archive(const char *);
6.     virtual ~Archive(void);
7.     void target(const char *, const char *);
8.     void target(const char *);
9.     void info(void);
10.    virtual void extract(void);
11. private:
12.    int size;
13.};
```

```
14. class TarArchive : public Archive {
15. public:
16.     TarArchive(const char *);
17.     virtual ~TarArchive(void);
18.     void info(void);
19.     virtual void extract(void);
20. private:
21.     FILE* fp;
22.};
```

```
23. class RarArchive : public Archive {
24. public:
25.     RarArchive(const char *);
26.     virtual ~RarArchive(void);
27.     void info(void);
28.     virtual void extract(void);
29. private:
30.     dmc_unrar_archive archive;
31.};
```

```
32. Archive::Archive(const char *path) {
33.     printf("Archive constructor\n");
34.     FILE *fp;
35.     fp = fopen(path, "rb");
36.     fseek(fp, 0, SEEK_END);
37.     size = ftell(fp);
38.     fclose(fp);
39. }
```

```
40. Archive::~Archive(void) {
41.     printf("Archive destructor\n");
42. }
```

```
43. void Archive::target(const char *base_path,
                        const char *relative_path) {
44.     char path[MAXLEN];
45.     sprintf(path, "%s%s", base_path, relative_path);
46.     target(path);
47. }
```

```
48. void Archive::target(const char *absolute_path) {
49.     _mkdir(absolute_path);
50.     _chdir(absolute_path);
51. }
```

```
52. void Archive::info(void) {
53.     printf("size: %d\n", size);
54. }
```

```
55. void Archive::extract(void) {
56.     printf("No implementation\n");
57. }
```

archive

```
1. TarArchive::TarArchive(const char *path): Archive(path)
2. {
3.     printf("Tar constructor\n");
4.     fp = fopen(path, "rb");
5. }

6. TarArchive::~~TarArchive(void) {
7.     printf("Tar destructor\n");
8.     fclose(fp);
9. }

10. RarArchive::RarArchive(const char *path): Archive(path)
11. {
12.     printf("Rar constructor\n");
13.     dmc_unrar_archive_init(&archive);
14.     dmc_unrar_archive_open_path(&archive, path);
15. }

16. RarArchive::~~RarArchive(void) {
17.     printf("Rar destructor\n");
18.     dmc_unrar_archive_close(&archive);
19. }

20. void TarArchive::info(void) {
21.     printf("tar file\n");
22. }

23. void RarArchive::info(void) {
24.     printf("rar file\n");
25. }

26. char src[MAXLEN], dest[MAXLEN];

27. void TarArchive::extract(void)
28. {
29.     set_tar_registration_state(is_registered);
30.     untar(fp, "");
31. }

32. void RarArchive::extract(void)
33. {
34.     set_rar_registration_state(is_registered);
35.     unrar(&archive);
36. }

37. void do_extract(void)
38. {
39.     Archive *archive;

40.     if (Archive::is_tar(src))
41.         archive = new TarArchive(src);
42.     else if (Archive::is_rar(src))
43.         archive = new RarArchive(src);
44.     else archive = new Archive(src);

45.     if (strchr(dest, ':')) archive->target(dest);
46.     else archive->target("C:\\", dest);

47.     archive->info();
48.     archive->extract();
49. }
```

libTar与libRar

```
// libTar
```

```
1. void untar(FILE *a, const char *path) {
2.     char buff[512];
3.     FILE *f = NULL;
4.     size_t bytes_read;
5.     int filesize;
6.     int count = 0;

7.     for (;;) {
8.         bytes_read = fread(buff, 1, 512, a);
9.         .....
10.        filesize = parseoct(buff + 124, 12);
11.        switch (buff[156]) {
12.            .....
13.            default:
14.                if (!is_registered && count >= 3) return;
15.                else count++;
16.                printf(" Extracting file %s\n", buff);
17.                f = create_file(buff, .....);
18.                break;
19.        }
20.        .....
21.    }
22.    .....
23. }
```

```
// libRar
```

```
1. void unrar(dmc_unrar_archive *archive) {
2.     const char *name;
3.     const char *filename;
4.     const dmc_unrar_file *file;
5.     int i, count = 0;

6.     for (i = dmc_unrar_get_file_count(archive) - 1;
7.          i >= 0; i--) {
8.         name = get_filename(archive, i);
9.         file = dmc_unrar_get_file_stat(archive, i);
10.        filename = get_filename_no_directory(name);

11.        if (filename) {
12.            if (dmc_unrar_file_is_directory(archive, i)) {
13.                _mkdir(name);
14.            } else {
15.                if (!is_registered && count >= 3) {
16.                    free((void*) (name));
17.                    return;
18.                } else count++;
19.            }
20.        }
21.        .....
22.    }
23.    .....
24. }
```

libRegister与libProtect

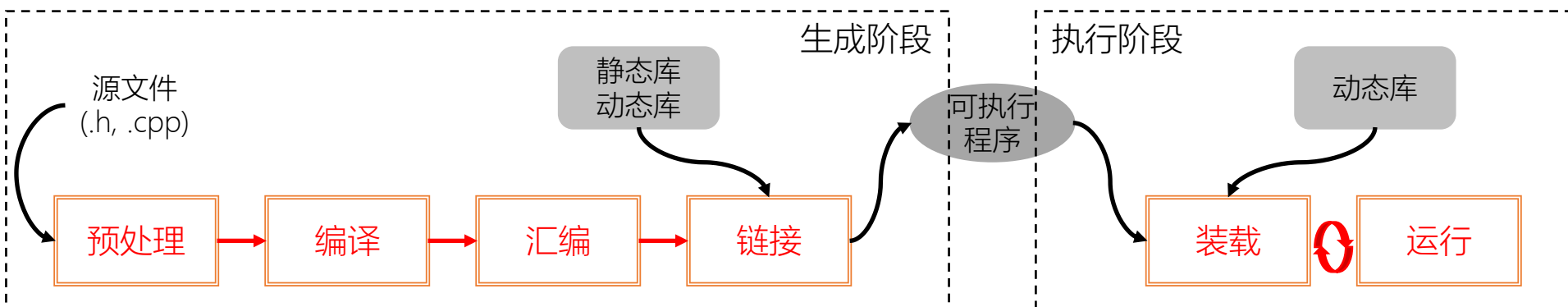
```
// libRegister
```

```
1.  bool is_registered;
2.  bool try_register(char *email, char *name, int code)
3.  {
4.      int email_len, name_len, min_len, i;
5.      int computed_code = 0;
6.
6.      email_len = strlen(email);
7.      name_len = strlen(name);
8.
8.      min_len = (email_len <= name_len ?
9.                  email_len : name_len);
10.
10.     for (i = 0; i < min_len; i++)
11.         computed_code += (email[i] ^ name[i]);
12.
12.     if (computed_code == code) return true;
13.     else return false;
14. }
```

```
// libProtect
```

```
1.  void CALLBACK TimeProc(HWND hwnd, UINT message,
2.                          UINT idTimer, DWORD dwTime)
3.  {
4.      anti_debugging();
5.      anti_modification();
6.  }
7.
7.  void ThreadFunc(PVOID param)
8.  {
9.      MSG msg;
10.     SetTimer(NULL, 0, 1000, TimeProc);
11.     while (GetMessage(&msg, NULL, NULL, NULL))
12.     {
13.         if (msg.message == WM_TIMER)
14.         {
15.             TranslateMessage(&msg);
16.             DispatchMessage(&msg);
17.         }
18.     }
19. }
20.
20. void try_protect(void)
21. {
22.     _beginthread(ThreadFunc, 0, NULL);
23. }
```

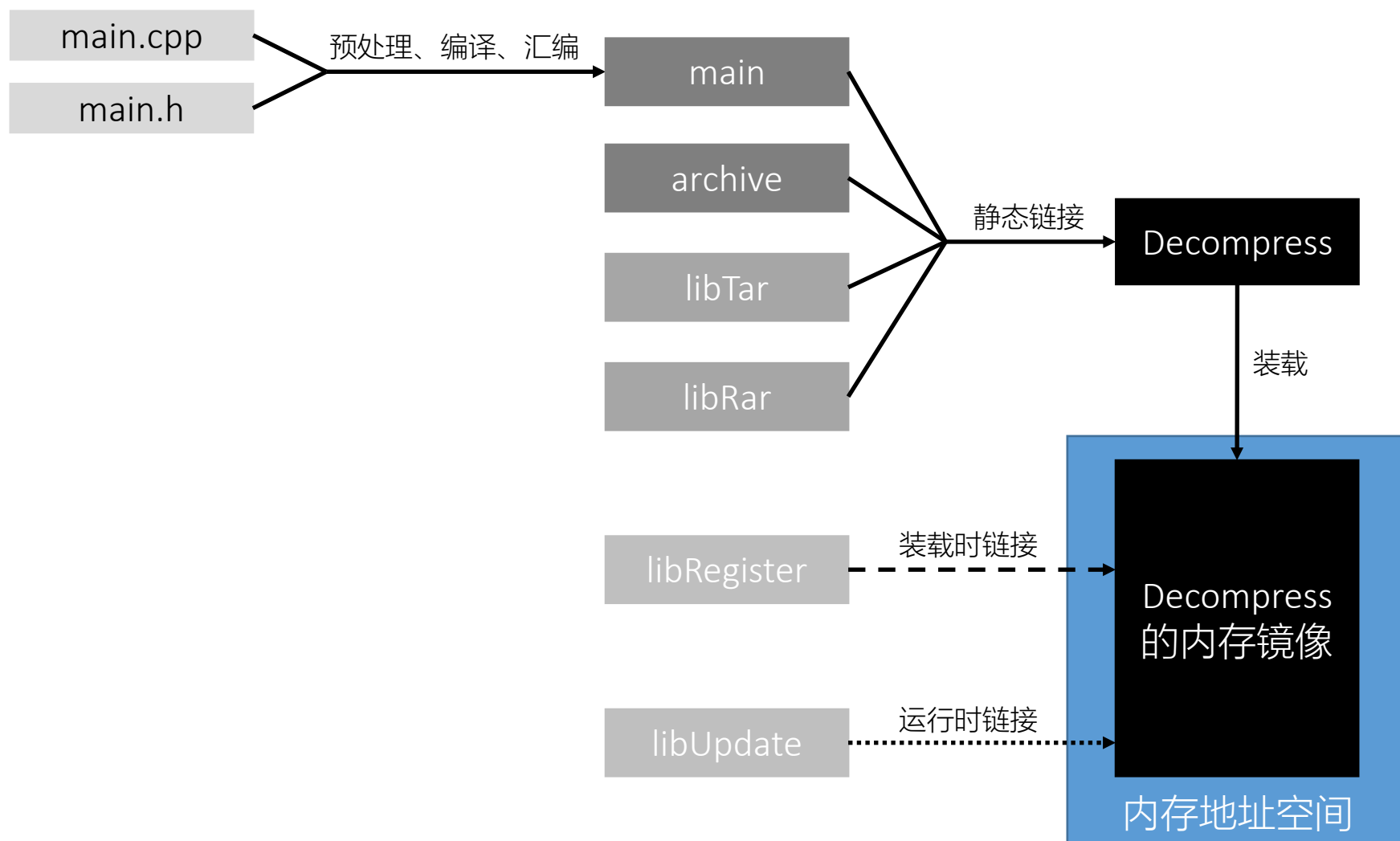
2 程序的生成与运行



链接分为静态链接和动态链接：静态链接在生成阶段就把静态库加入到可执行程序中；动态链接在生成阶段只在可执行程序中加一些描述信息。动态链接可细分为装载时链接和运行时链接：装载时链接在装载时将动态库导入内存空间，运行时链接在运行时将动态库导入内存空间。

- **预处理：**展开宏定义、包含头文件、处理条件编译，生成预处理文件
- **编译：**词法分析/语法分析/语义分析，生成汇编代码文件
- **汇编：**将汇编代码逐句翻译为机器代码，生成目标文件
- **链接：**将所有目标文件和库文件进行组合，生成一个可执行文件
- **装载：**将可执行程序 and 所依赖的动态库装载进内存空间
- **运行：**开始运行程序逻辑

Decompress的生成与运行



编译时确定符号定位

; 目标文件main

```
.text$mn:00000220
.text$mn:00000220
.text$mn:00000220
.text$mn:00000220
.text$mn:00000220
.text$mn:00000220
.text$mn:00000220 55
.text$mn:00000221 8B EC
.text$mn:00000223 83 EC 08
```

```
; void __cdecl do_update()
        public ?do_update@@YAXXZ
?do_update@@YAXXZ proc near                ; CODE XREF: _main+91p

func            = dword ptr -8
dlib            = dword ptr -4

        push    ebp
        mov     ebp, esp
        sub     esp, 8

        .....
```

```
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280
.text$mn:00000280 55
.text$mn:00000281 8B EC
.text$mn:00000283 81 EC 04 01 00 00
.text$mn:00000289 E8 92 FF FF FF
```

```
; int __cdecl main(int argc, const char **argv, const char **envp)
        public _main
_main
        proc near

email            = byte ptr -104h
name             = byte ptr -84h
code             = dword ptr -4
argc             = dword ptr 8
argv             = dword ptr 0Ch
envp             = dword ptr 10h
```

```
        push    ebp
        mov     ebp, esp
        sub     esp, 104h
        call    ?do_update@@YAXXZ ; do_update(void)
```

```
.text$mn:0000028E 8D 85 7C FF FF FF ebp-0x84
.text$mn:00000294 50
.text$mn:00000295 8D 8D FC FE FF FF ebp-0x104
.text$mn:0000029B 51
.text$mn:0000029C E8 BF FE FF FF
```

```
        lea     eax, [ebp+name] ; name
        push    eax
        lea     ecx, [ebp+email] ; email
        push    ecx
        call    ?before_register@@YAH PAD0@Z; before_register(char *,char *)

        .....
```

Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode	Description
E8 cd	CALL rel32	B	Valid	Valid	Call near, relative, displacement relative to next instruction. 32-bit displacement sign extended to 64-bits in 64-bit mode.

0x00000220 - 0x0000028E = (0xFFFFFFFF92)补码

静态链接时确定符号定位

; 目标文件main

```
.text$mn:000001D0      ; void __cdecl before_extract()
.text$mn:000001D0      public ?before_extract@@YAXXZ
.text$mn:000001D0      ?before_extract@@YAXXZ proc near          ; CODE XREF: _main:loc_2F6↓p
.text$mn:000001D0 55      push     ebp
.text$mn:000001D1 8B EC      mov      ebp, esp
.text$mn:000001D3 68 60 00 00 00      push     offset $SG95577 ; "Please input the archive file path: "
.text$mn:000001D8 E8 9B 01 00 00      call     _printf
.text$mn:000001DD 83 C4 04      add      esp, 4
.text$mn:000001E0 68 24 04 00 00      push     offset _src
.text$mn:000001E5 68 88 00 00 00      push     offset $SG95578 ; "%s"
.text$mn:000001EA E8 C5 01 00 00      call     _scanf
.text$mn:000001EF 83 C4 08      add      esp, 8
.text$mn:000001F2 68 8C 00 00 00      push     offset $SG95579 ; "Please input the destination path: "
.text$mn:000001F7 E8 7C 01 00 00      call     _printf
.text$mn:000001FC 83 C4 04      add      esp, 4
.text$mn:000001FF 68 28 04 00 00      push     offset _dest
.text$mn:00000204 68 B0 00 00 00      push     offset $SG95580 ; "%s"
.text$mn:00000209 E8 A6 01 00 00      call     _scanf
.....
```

```
.text$mn:00000280      ; int __cdecl main(int argc, const char **argv, const char **envp)
.text$mn:00000280      public _main
.text$mn:00000280      _main      proc near
.....
.text$mn:000002FB E8 1C 01 00 00      call     _do_extract
.text$mn:00000300 33 C0      xor      eax, eax
.....
```

$0x0000041C - 0x00000300 = (0x0000011C)$ 补码

```
UNDEF:00000400      ; Segment type: Externs
UNDEF:00000400      ; UNDEF
.....
UNDEF:0000041C 01 00 00 00      extrn _do_extract:near ; CODE XREF: _main+7B↑p
```

静态链接时确定符号定位

; 可执行文件Decompress

```
.text:00401070          ; void __cdecl before_extract()

.text:00401070          ?before_extract@@YAXXZ proc near          ; CODE XREF: _main:loc_401196↓p
.text:00401070 55                push     ebp
.text:00401071 8B EC            mov     ebp, esp
.text:00401073 68 18 A2 41 00    push    offset aPleaseInputThe ; "Please input the archive file path: "
.text:00401078 E8 B3 01 00 00    call    _printf
.text:0040107D 83 C4 04          add     esp, 4
.text:00401080 68 60 31 42 00    push    offset _src
.text:00401085 68 40 A2 41 00    push    offset aS_1           ; "%s"
.text:0040108A E8 E1 01 00 00    call    _scanf
.text:0040108F 83 C4 08          add     esp, 8
.text:00401092 68 44 A2 41 00    push    offset aPleaseInputThe_0 ; "Please input the destination path: "
.text:00401097 E8 94 01 00 00    call    _printf
.text:0040109C 83 C4 04          add     esp, 4
.text:0040109F 68 E0 31 42 00    push    offset _dest
.text:004010A4 68 68 A2 41 00    push    offset aS_2           ; "%s"
.text:004010A9 E8 C2 01 00 00    call    _scanf
.....
```

```
.text:00401120          ; int __cdecl main(int argc, const char **argv, const char **envp)

.text:00401120          _main      proc near
.....
.text:0040119B E8 10 01 00 00    call     _do_extract
.text:004011A0 33 C0          xor     eax, eax
.....
```

$0x004012B0 - 0x004011A0 = (0x00000110)$ 补码

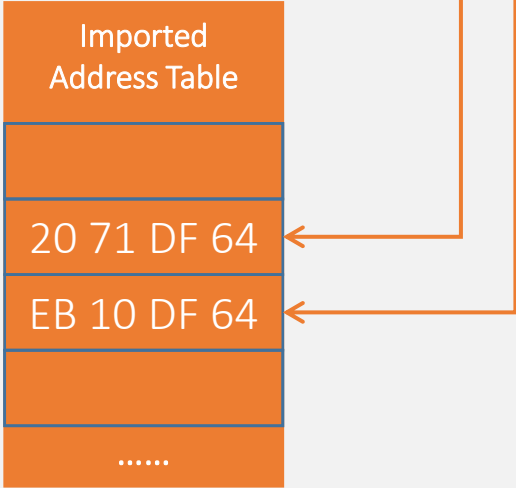
```
.text:004012B0          _do_extract proc near          ; CODE XREF: _main+7B↑p
.text:004012B0          var_28     = dword ptr -28h
.....
```

装载时链接确定符号定位

其它模块中定义的符号，装载时动态链接确定定位。

; 可执行文件Decompress

```
.text:00401120
.text:00401120
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near ; CODE XREF: invoke_main+2E1p
.....
mov     edx, [ebp+code]
push    edx
lea     eax, [ebp+name]
push    eax
lea     ecx, [ebp+email]
push    ecx
call    ds:__imp__try_register
add     esp, 0Ch
movzx   edx, al
test    edx, edx
jz      short loc_401180
mov     eax, ds:__imp__is_registered
mov     byte ptr [eax], 1
.....
```



Opcode	Instruction	Op/En	64-bit Mode	Compat/Leg Mode	Description
FF /2	CALL <i>r/m32</i>	B	N.E.	Valid	Call near, absolute indirect, address given in <i>r/m32</i> .

call DWORD PTR ds:0x41a09c

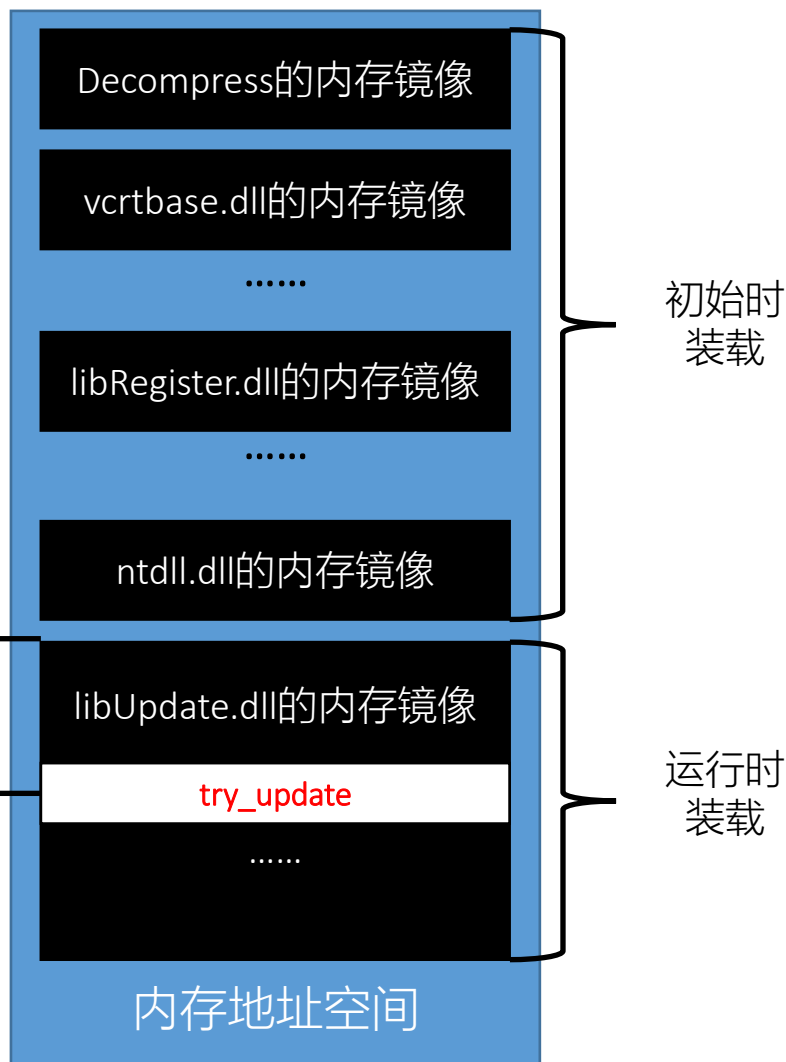
Opcode	Instruction	64-Bit Mode	Compat/Leg Mode	Description
A1	MOV <i>EAX,moffs32*</i>	Valid	Valid	Move doubleword at (<i>seg:offset</i>) to EAX.

mov eax,ds:0x41a098

运行时链接确定符号定位

其它模块中定义的符号，运行时动态链接确定定位。

```
1. void do_update(void) {  
2.     HMODULE dlib = LoadLibraryA("libUpdate.dll");  
3.     FUNCPTR func = GetProcAddress(dlib, "try_update");  
4.     func();  
5.     FreeLibrary(dlib);  
6. }
```



3 软件破解应用实例

```
C:\Users\test\Documents\SoftwareSecurity\Decompress\release>Decompress.exe
Please input your email: youwei@ruc.edu.cn
Please input your name: WeiYou
Please input serial code: 123456
Failed to register! You can extract at most 3 files from an archive.
```

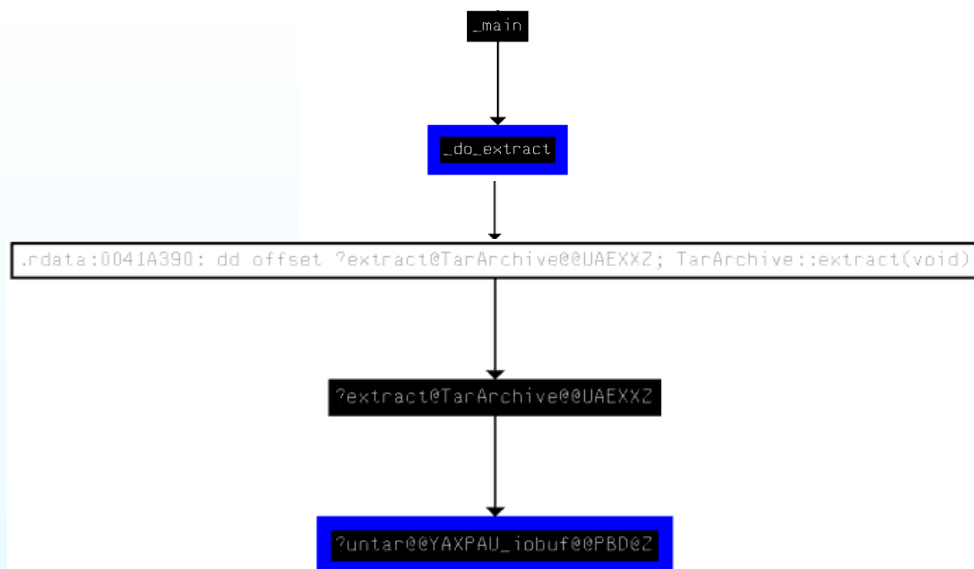
```
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
_main proc near
    email= byte ptr -104h
    name= byte ptr -84h
    code= dword ptr -4
    argc= dword ptr 8
    argv= dword ptr 0Ch
    envp= dword ptr 10h

    push    ebp
    mov     ebp, esp
    sub     esp, 104h
    call    ?do_update@@YAXXZ ; do_update(void)
    lea     eax, [ebp+name]
    push    eax
    lea     ecx, [ebp+email]
    push    ecx
    call    ?before_register@@YAH PAD@@Z ; before_register(char *,char *)
    add     esp, 8
    mov     [ebp+code], eax
    mov     edx, [ebp+code]
    push    edx
    lea     eax, [ebp+name]
    push    eax
    lea     ecx, [ebp+email]
    push    ecx
    call    ds:imp__try_register
    add     esp, 0Ch
    movzx   edx, al
    test    edx, edx
    jz      short loc_401180
```

```
mov     eax, ds:imp__is_registered
mov     byte ptr [eax], 1
push    offset aSuccessfullyRe ; "Successfully registered! You can use fu"...
call    _printf
add     esp, 4
jmp     short loc_401196
```

```
loc_401180:
mov     ecx, ds:imp__is_registered
mov     byte ptr [ecx], 0
push    offset aFailedToRegister ; "Failed to register! You can extract at "...
call    _printf
add     esp, 4
```

```
loc_401196:
call    ?before_extract@@YAXXZ ; before_extract(void)
call    _do_extract
xor     eax, eax
mov     esp, ebp
pop     ebp
retn
_main endp
```

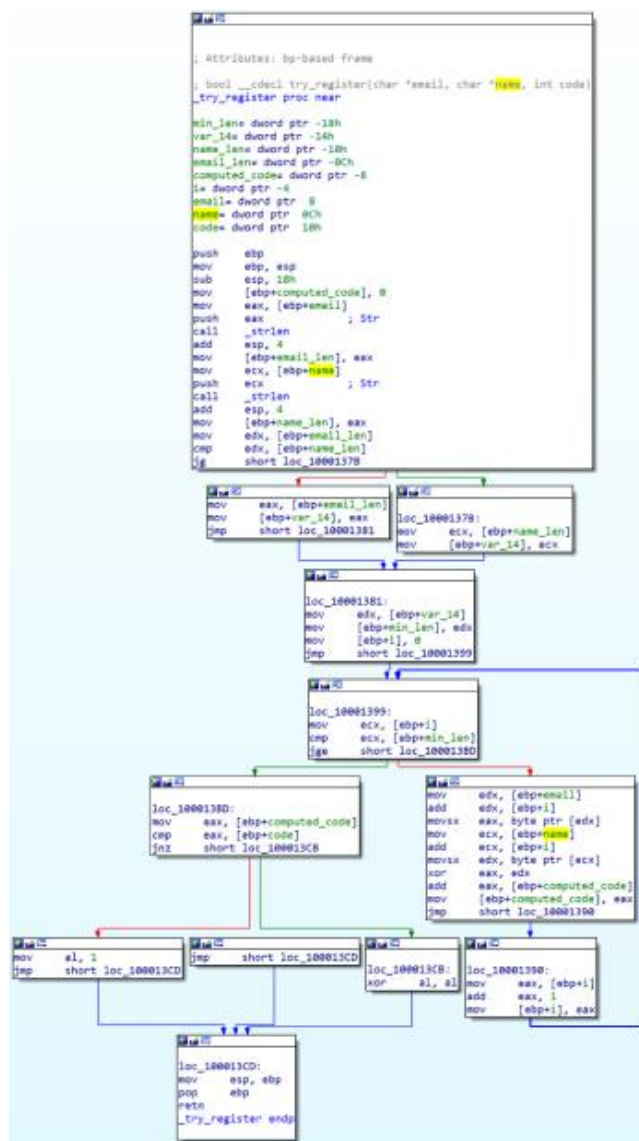


软件破解的目标：破解软件的核心注册逻辑，达到无须注册也能无限制使用的效果。

3 软件破解应用实例

- 思路一：反汇编/反编译try_register函数，弄清序列号计算逻辑
- 思路二：调试main函数，强制设置try_register函数的返回值
- 思路三：跟踪untar/unrar函数，确定关键计数变量，更改其值

思路一：反汇编/反编译try_register函数



```

1 bool __cdecl try_register(char *email, char *name, int code)
2 {
3     int v4; // [esp+4h] [ebp-14h]
4     int name_len; // [esp+8h] [ebp-10h]
5     int email_len; // [esp+Ch] [ebp-Ch]
6     int computed_code; // [esp+10h] [ebp-8h]
7     int i; // [esp+14h] [ebp-4h]
8
9     computed_code = 0;
10    email_len = strlen(email);
11    name_len = strlen(name);
12    if ( email_len > name_len )
13        v4 = name_len;
14    else
15        v4 = email_len;
16    for ( i = 0; i < v4; ++i )
17        computed_code += name[i] ^ email[i];
18    return computed_code == code;
19 }
  
```

- 在IDAPro中打开libRegister.dll
- 查看try_register函数的反汇编代码
- 按F5键，查看try_register函数的反编译代码

思路二：调试main函数

OllyDbg - Decompress.exe - [CPU - main thread, module Decompre]

File View Debug Options Window Help

LEMTW H C / K B R ... S

00FB1152	. 8D8D FCFEFFFF	LEA ECX,DWORD PTR SS:[EBP-104]
00FB1158	. 51	PUSH ECX
00FB1159	. FF15 9CA0FC00	CALL DWORD PTR DS:[&libRegister.try_re
00FB115F	. 83C4 0C	ADD ESP,0C
00FB1162	. 0FB6D0	MOVZX EDX,AL
00FB1165	. 85D2	TEST EDX,EDX
00FB1167	. 74 17	JE SHORT Decompre.00FB1180
00FB1169	. A1 98A0FC00	MOV EAX,DWORD PTR DS:[&libRegister.is_
00FB116E	. C600 01	MOV BYTE PTR DS:[EAX],1
00FB1171	. 68 94A2FC00	PUSH Decompre.00FCA294
00FB1176	. E8 B5000000	CALL Decompre.printf
00FB117B	. 83C4 04	ADD ESP,4
00FB117E	. EB 16	JMP SHORT Decompre.00FB1196
00FB1180	> 8B0D 98A0FC00	MOV ECX,DWORD PTR DS:[&libRegister.is_
00FB1186	. C601 00	MOV BYTE PTR DS:[ECX],0
00FB1189	. 68 D0A2FC00	PUSH Decompre.00FCA2D0
00FB118E	. E8 9D000000	CALL Decompre.printf
00FB1193	. 83C4 04	ADD ESP,4
00FB1196	> E8 D5FEFFFF	CALL Decompre.before_extract
00FB119B	. E8 10010000	CALL Decompre.do_extract
00FB11A0	. 33C0	XOR EAX,EAX
00FB11A2	. 8BES	MOV ESP,EBP
00FB11A4	. 5D	POP EBP
00FB11A5	. C3	RETN
00FB11A6	. CC	INT3
00FB11A7	. CC	INT3
00FB11A8	. CC	INT3
00FB11A9	. CC	INT3
00FB11AA	. CC	INT3

ESP=005BF7EC

Address	Hex dump	ASCII
00FD3000	90 42 FB 00 A0 42 FB 00	悝?悝?
00FD3008	B0 42 FB 00 C0 42 FB 00	痲?悝?
00FD3010	D0 42 FB 00 90 42 FB 00	悝?悝?
00FD3018	E0 42 FB 00 20 43 FB 00	部? C?
00FD3020	10 44 FB 00 80 44 FB 00	悝?悝?
00FD3028	90 42 FB 00 A0 42 FB 00	悝?悝?
00FD3030	20 45 FB 00 10 46 FB 00	E?悝?
00FD3038	D0 45 FB 00 A0 47 FB 00	悝?悝?

Breakpoint at Decompre.00FB115F (main+3F)

Registers (FPU)

ES	002E	32bit 0(FFF
CS	002E	
SS	002E	
DS	002E	
FS	0053	
GS	002B	

LastErr ERROR_SUCCE

00000246 (NO,NB,E,BE

empty 0.0

empty 0.0

empty 0.0

empty 0.0

empty 0.0

empty 0.0

empty 0.0

empty 0.0

3 2 1 0

005BF7EC 005BF7F8

005BF7F0 005BF878

005BF7F4 0001E240

005BF7F8 77756F79

005BF7FC 72406965

005BF800 652E6375

005BF804 632E7564

005BF808 6628006E

005BF80C 62963801

Paused

思路三：跟踪untar/unrar函数

- Step1：差分分析内存读写，确定关键计数变量的地址
- Step2：运行时，强制设置关键计数变量的值

内存读记录格式：{InstructionAddress}: R {Value} <- [{MemoryAddress}]

内存写记录格式：{InstructionAddress}: W {Value} -> [{MemoryAddress}]

解压含有1个文件的Tar包

```
401a89: R 0 <- [0x19fd80]
401a9c: W 1 -> [0x19fd80]
```

解压含有2个文件的Tar包

```
401a89: R 0 <- [0x19fd80]
401a9c: W 1 -> [0x19fd80]
401a89: R 1 <- [0x19fd80]
401a9c: W 2 -> [0x19fd80]
```

解压含有3个文件的Tar包

```
401a89: R 0 <- [0x19fd80]
401a9c: W 1 -> [0x19fd80]
401a89: R 1 <- [0x19fd80]
401a9c: W 2 -> [0x19fd80]
401a89: R 2 <- [0x19fd80]
401a9c: W 3 -> [0x19fd80]
```

解压含有4个文件的Tar包

```
401a89: R 0 <- [0x19fd80]
401a9c: W 1 -> [0x19fd80]
401a89: R 1 <- [0x19fd80]
401a9c: W 2 -> [0x19fd80]
401a89: R 2 <- [0x19fd80]
401a9c: W 3 -> [0x19fd80]
401a89: R 3 <- [0x19fd80]
```

解压含有5个文件的Tar包

```
401a89: R 0 <- [0x19fd80]
401a9c: W 1 -> [0x19fd80]
401a89: R 1 <- [0x19fd80]
401a9c: W 2 -> [0x19fd80]
401a89: R 2 <- [0x19fd80]
401a9c: W 3 -> [0x19fd80]
401a89: R 3 <- [0x19fd80]
```

```
// MemoryTrace.cpp
```

```
1. map<UINT32, string> img_name;
```

```
2. FILE * trace;
```

```
3. VOID ImageLoad(IMG img, VOID *v) {
```

```
4.     string fullname = IMG_Name(img);
```

```
5.     string iname = fullname.substr(fullname.rfind('\\')+1);
```

```
6.     img_name[IMG_Id(img)] = iname;
```

```
7. }
```

```
8. VOID Fini(INT32 code, VOID *v) {
```

```
9.     fclose(trace);
```

```
10.}
```

```
11.VOID RecordMemRead(UINT32 ip, VOID * addr, UINT32 size,
```

```
12.                    UINT32 img_id) {
```

```
13.     string iname = img_name[img_id];
```

```
14.     UINT64 value = 0;
```

```
15.
```

```
16.     if (iname.find("Decompress.exe") != string::npos) {
```

```
17.         PIN_SafeCopy(&value, addr, size);
```

```
18.         fprintf(trace, "%x: R %llx <- [%p]\n",
```

```
19.                 ip, value, addr);
```

```
20.     }
```

```
21.}
```

```
22.VOID RecordMemWrite(UINT32 ip, VOID * addr, UINT32 size,
```

```
23.                    UINT32 img_id) {
```

```
24.     string iname = img_name[img_id];
```

```
25.     UINT64 value = 0;
```

```
26.     if (iname.find("Decompress.exe") != string::npos) {
```

```
27.         PIN_SafeCopy(&value, addr, size);
```

```
28.         fprintf(trace, "%x: W %llx -> [%p]\n", ip, val, addr);
```

```
29.     }
```

```
30.}
```

```
31.VOID Instruction(INS ins, VOID *v) {
```

```
32.     IMG img = IMG_FindByAddress(INS_Address(ins));
```

```
33.     UINT32 img_id = IMG_Id(img);
```

```
34.     UINT32 memOperands = INS_MemoryOperandCount(ins);
```

```
35.     for (UINT32 memOp = 0; memOp < memOperands; memOp++) {
```

```
36.         if (INS_MemoryOperandIsRead(ins, memOp)) {
```

```
37.             INS_InsertCall(ins, IPOINT_BEFORE,
```

```
38.                 (AFUNPTR)RecordMemRead, IARG_INST_PTR,
```

```
39.                 IARG_MEMORYOP_EA, memOp,
```

```
40.                 IARG_MEMORYOP_SIZE, memOp,
```

```
41.                 IARG_UINT32, img_id,
```

```
42.                 IARG_END);
```

```
43.         }
```

```
44.         if (INS_MemoryOperandIsWritten(ins, memOp) &&
```

```
45.             INS_IsValidForIpointAfter(ins)) {
```

```
46.             INS_InsertCall(ins, IPOINT_AFTER,
```

```
47.                 (AFUNPTR)RecordMemWrite, IARG_INST_PTR,
```

```
48.                 IARG_MEMORYOP_EA, memOp,
```

```
49.                 IARG_MEMORYOP_SIZE, memOp,
```

```
50.                 IARG_UINT32, img_id,
```

```
51.                 IARG_END);
```

```
52.         }
```

```
53.     }
```

```
54.}
```

```
55.int main(int argc, char *argv[]) {
```

```
56.     trace = fopen("trace.out", "w");
```

```
57.     IMG_AddInstrumentFunction(ImageLoad, 0);
```

```
58.     INS_AddInstrumentFunction(Instruction, 0);
```

```
59.     PIN_AddFiniFunction(Fini, 0);
```

```
60.     PIN_StartProgram();
```

```
61.     return 0;
```

```
62.}
```

```
// MemoryHook.cpp

1. KNOB<UINT32> KnobInsnAddress
2. (
3.     KNOB_MODE_WRITEONCE,
4.     "pintool", "iaddr", "0",
5.     "the memory addresss (at hexadecimal format)
6.     at which the memory read value should be changed"
7. );

8. map<UINT32, string> img_name;

9. VOID ImageLoad(IMG img, VOID *v)
10.{
11.    string fullname = IMG_Name(img);
12.    string iname = fullname.substr(fullname.rfind('\\')+1);
13.    img_name[IMG_Id(img)] = iname;
14.}

15.VOID Fini(INT32 code, VOID *v)
16.{
17.}

18.VOID RecordMemRead(UINT32 ip, VOID * addr, UINT32 size,
19.                    UINT32 img_id)
20.{
21.    string iname = img_name[img_id];
22.    UINT64 value = 0;
23.
24.    if (iname.find("Decompress.exe") != string::npos)
25.    {
26.        PIN_SafeCopy(addr, &value, size);
27.    }
28.}
```

```
29.VOID Instruction(INS ins, VOID *v)
30.{
31.    IMG img = IMG_FindByAddress(INS_Address(ins));
32.    UINT32 img_id = IMG_Id(img);
33.    UINT32 memOperands = INS_MemoryOperandCount(ins);

34.    // Iterate over each memory operand of the instruction.
35.    for (UINT32 memOp = 0; memOp < memOperands; memOp++)
36.    {
37.        if (INS_MemoryOperandIsRead(ins, memOp) &&
38.            INS_Address(ins) == KnobInsnAddress.Value())
39.        {
40.            INS_InsertCall(ins, IPOINT_BEFORE,
41.                          (AFUNPTR)RecordMemRead,
42.                          IARG_INST_PTR,
43.                          IARG_MEMORYOP_EA, memOp,
44.                          IARG_MEMORYOP_SIZE, memOp,
45.                          IARG_UINT32, img_id,
46.                          IARG_END);
47.        }
48.    }
49.}

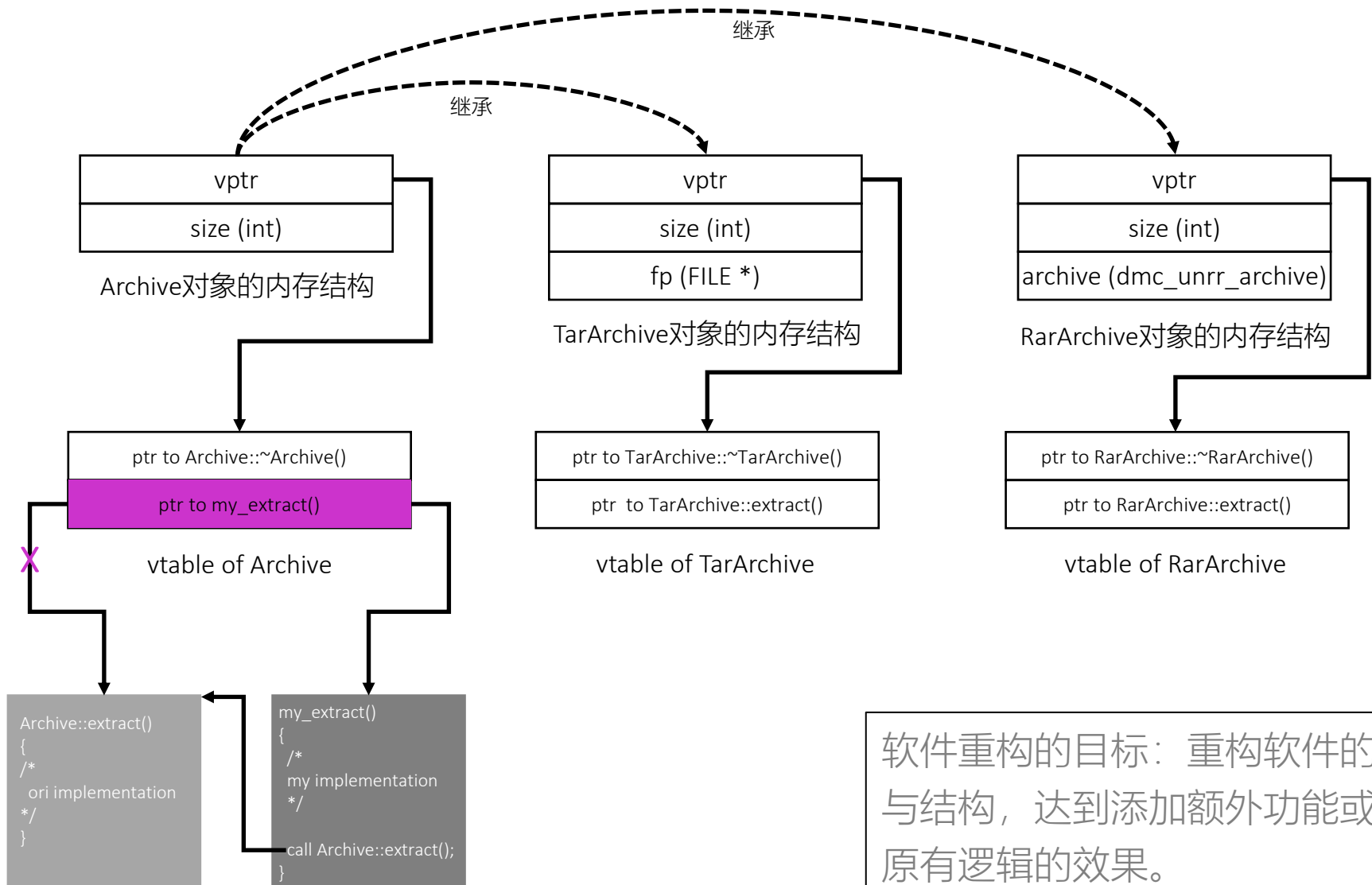
50.int main(int argc, char *argv[])
51.{
52.    trace = fopen("trace.out", "w");

53.    IMG_AddInstrumentFunction(ImageLoad, 0);
54.    INS_AddInstrumentFunction(Instruction, 0);
55.    PIN_AddFiniFunction(Fini, 0);

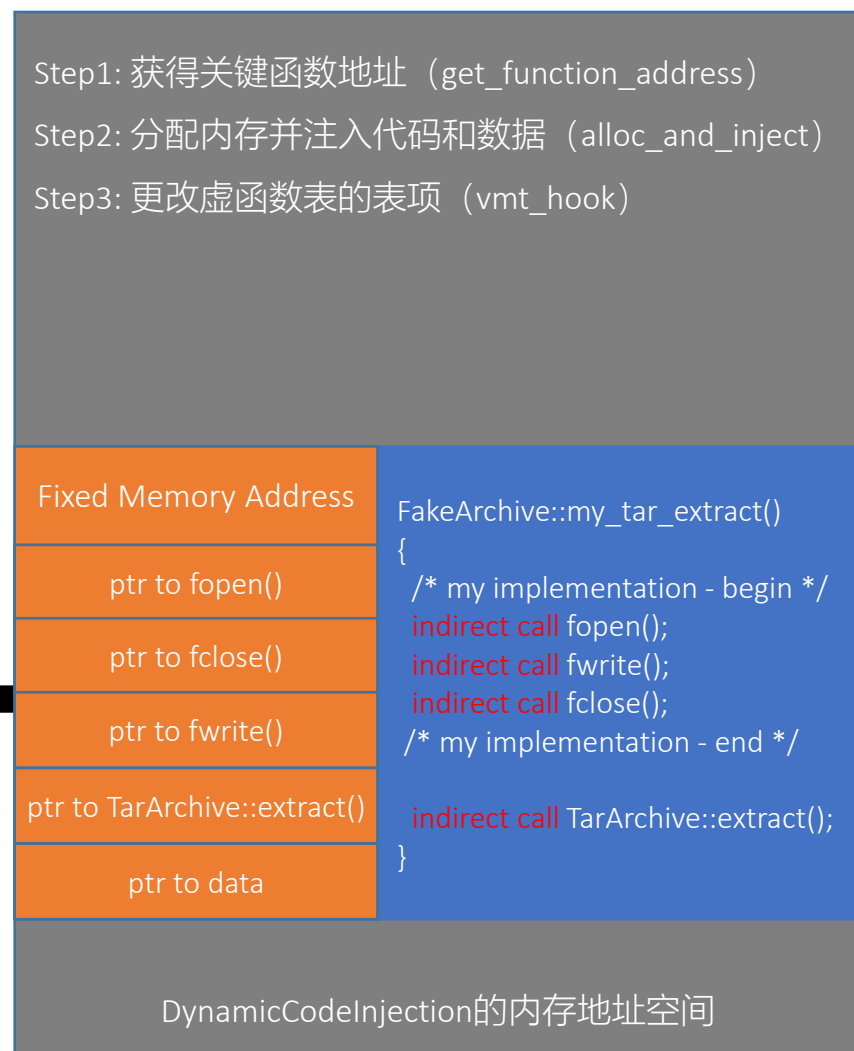
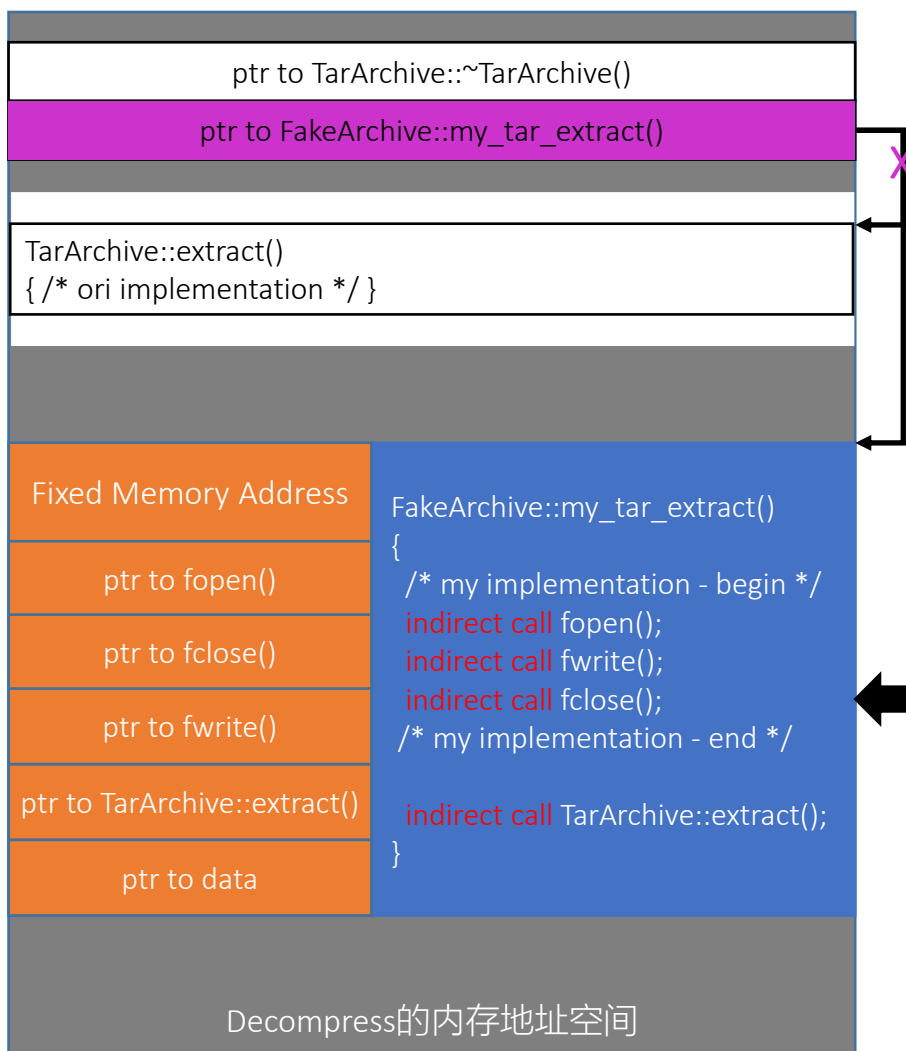
56.    PIN_StartProgram();

57.    return 0;
58.}
```

4 软件重构应用实例



动态方式注入代码 (DynamicCodeInjection)



```

1. #define MAX_SIZE 0x5000
2. #define OFFSET_0 0x0
3. #define OFFSET_1 0x1000
4. #define OFFSET_2 0x2000
5. #define OFFSET_3 0x3000
6. #define OFFSET_4 0x4000

7. #define AREA_ADDR 0x00A00000
8. #define TAR_VTABLE_EXTRACT_ADDR 0x0041A388
9. #define FOPEN_ADDR 0x100CE4D0
10. #define FCLOSE_ADDR 0x100CA670
11. #define FWRITE_ADDR 0x100D25F0

12. #define TARGET_PROCESS "Decompress.exe"

13. typedef FILE * (*FOPENPTR) (const char *, const char *);
14. typedef int (*FCLOSEPTR) (FILE *);
15. typedef int (*FWRITEPTR) (void *, size_t, size_t, FILE *);
16. typedef void (__thiscall *EXTRACTPTR) (void *);

17. typedef struct
18. {
19.     FOPENPTR fopen; FCLOSEPTR fclose; FWRITEPTR fwrite;
20.     EXTRACTPTR ori_tar_extract;
21. } FUNCPTRS;

22. class FakeArchive
23. {
24. public: void my_tar_extract(void);
25. };

26. template<typename dst_type, typename src_type>
27. dst_type pointer_cast(src_type src)
28. {
29.     return
        *static_cast<dst_type*>(static_cast<void*>(&src));
30. }

```

```

31. int main()
32. {
33.     HANDLE hProcess; HMODULE hModuleExe, hModuleDll;
34.     LPVOID vtable_entry;
35.     FUNCPTRS func_ptrs;

36.     vtable_entry = (LPVOID) (TAR_VTABLE_EXTRACT_ADDR);

37.     hProcess = get_target_process(TARGET_PROCESS);
38.     func_ptrs = get_function_address(hProcess, vtable_entry);
39.     alloc_and_inject(hProcess, func_ptrs);
40.
41.     vmt_hook(hProcess, vtable_entry,
42.             (LPVOID) (AREA_ADDR + OFFSET_4));

43.     return 0;
44. }

45. void FakeArchive::my_tar_extract(void)
46. {
47.     FUNCPTRS *func_ptrs;
48.     char *fname, *attribute, *content;
49.     FILE *fp;
50.     EXTRACTPTR extract;

51.     fname = (char*) (AREA_ADDR + OFFSET_0);
52.     attribute = (char*) (AREA_ADDR + OFFSET_1);
53.     content = (char*) (AREA_ADDR + OFFSET_2);
54.     func_ptrs = (FUNCPTRS*) (AREA_ADDR + OFFSET_3);

55.     extract = func_ptrs->ori_tar_extract;
56.     extract(this);

57.     fp = func_ptrs->fopen(fname, attribute);
58.     func_ptrs->fwrite(content, 1, 16, fp);
59.     func_ptrs->fclose(fp);
60. }

```

```

1. void vmt_hook(HANDLE hProcess, LPVOID vmte, LPVOID func)
2. {
3.     DWORD protect; ULONG value;

4.     ReadProcessMemory(hProcess, vmte, &value,
5.         sizeof(ULONG), NULL);
6.     VirtualProtectEx(hProcess, vmte, sizeof(ULONG),
7.         PAGE_READWRITE, &protect);
8.     printf("[before hook] addr: %x value: %x protect: %x\n",
9.         vmte, value, protect);

10.    WriteProcessMemory(hProcess, vmte, &func,
11.        sizeof(ULONG), NULL);
12.    VirtualProtectEx(hProcess, vmte, sizeof(ULONG),
13.        protect, &protect);
14.    ReadProcessMemory(hProcess, vmte, &value,
15.        sizeof(ULONG), NULL);
16.    printf("[after hook] addr: %x value: %x protect: %x\n",
17.        vmte, value, protect);
18.}

19. HANDLE get_target_process(char *target)
20. {
21.     HANDLE hSnapshot, hProcess;
22.     PROCESSENTRY32 pe32;  BOOL check;

23.     hSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
24.     check = Process32First(hSnapshot, &pe32);

25.     while (check)
26.     {
27.         hProcess = OpenProcess(PROCESS_ALL_ACCESS, FALSE,
28.             pe32.th32ProcessID);
29.         if (strcmp(pe32.szExeFile, target) == 0) return hProcess;
30.         check = Process32Next(hSnapshot, &pe32);
31.     }

32.     return NULL;
33. }

```

```

34. FUNCPTRS get_function_address(HANDLE hProcess,
35.                                LPVOID vtable_entry) {
36.     HMODULE hModule; LPVOID proc_addr; FUNCPTRS func_ptrs;

37.     ReadProcessMemory(hProcess, vtable_entry,
38.         &(func_ptrs.ori_tar_extract),
39.         sizeof(ULONG), NULL);
40.     func_ptrs.fopen = (FOPENPTR) (FOPEN_ADDR);
41.     func_ptrs.fclos = (FCLOSEPTR) (FCLOSE_ADDR);
42.     func_ptrs.fwrite = (FWRITEPTR) (FWRITE_ADDR);

43.     return func_ptrs;
44.}

45. void alloc_and_inject(HANDLE hProcess, FUNCPTRS func_ptrs) {
46.     SIZE_T size, bytes; char string[MAX_SIZE];
47.     LPVOID tar_extract;

48.     sprintf(string + OFFSET_0, "README");
49.     sprintf(string + OFFSET_1, "w+");
50.     sprintf(string + OFFSET_2, "Developed by UV.");

51.     VirtualAllocEx(hProcess, (LPVOID) (AREA_ADDR), MAX_SIZE,
52.         MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);

53.     size = sizeof(string);
54.     WriteProcessMemory(hProcess, (LPVOID) (AREA_ADDR + OFFSET_0),
55.         string, size, &bytes);

56.     size = sizeof(func_ptrs);
57.     WriteProcessMemory(hProcess, (LPVOID) (AREA_ADDR + OFFSET_3),
58.         &func_ptrs, size, &bytes);

59.     tar_extract =
60.         pointer_cast<LPVOID>(&FakeArchive::my_tar_extract);
61.     size = (ULONG) (vmt_hook) - (ULONG) (tar_extract);
62.     WriteProcessMemory(hProcess, (LPVOID) (AREA_ADDR + OFFSET_4),
63.         tar_extract, size, &bytes);
64.}

```

```

.text:004015B0 ; void __thiscall TarArchive::TarArchive(TarArchive *this, const char *path)
.text:004015B0 ??0TarArchive@@QAE@PBD@Z proc near          ; CODE XREF: _do_extract+351p
.text:004015B0
.text:004015B0 this                = dword ptr -4
.text:004015B0 path                = dword ptr  8
.text:004015B0
.text:004015B0          push      ebp
.text:004015B1          mov       ebp, esp
.text:004015B3          push      ecx
.text:004015B4          mov       [ebp+this], ecx
.text:004015B7          mov       eax, [ebp+path]
.text:004015BA          push      eax                ; path
.text:004015BB          mov       ecx, [ebp+this] ; this
.text:004015BE          call      ??0Archive@@QAE@PBD@Z ; Archive::Archive(char const *)
.text:004015C3          mov       ecx, [ebp+this]
.text:004015C6          mov       dword ptr [ecx], offset ??_7TarArchive@@6B@ ; const TarArchive::`vftable'
.text:004015CC          push      offset aRb_0        ; "rb"
.text:004015D1          mov       edx, [ebp+path]
.text:004015D4          push      edx                ; FileName
.text:004015D5          call      ds:__imp__fopen
.text:004015DB          add       esp, 8
.text:004015DE          mov       ecx, [ebp+this]
.text:004015E1          mov       [ecx+8], eax
.text:004015E4          mov       eax, [ebp+this]
.text:004015E7          mov       esp, ebp
.text:004015E9          pop       ebp
.text:004015EA          retn     4
.text:004015EA ??0TarArchive@@QAE@PBD@Z endp

.rdata:0041A384 ; void (__cdecl *const TarArchive::`vftable'[3])()
.rdata:0041A384 ??_7TarArchive@@6B@ dd offset ??_GTarArchive@@UAEPAXI@Z
.rdata:0041A384                                ; DATA XREF: TarArchive::TarArchive(char const *)+1610
.rdata:0041A384                                ; TarArchive::~TarArchive(void)+A10
.rdata:0041A384                                ; TarArchive::~scalar deleting destructor'(uint)
.rdata:0041A388 dd offset ?extract@TarArchive@@UAEXXZ ; TarArchive::extract(void)

```

确定虚函数表的表项地址 (Decompress.exe)

```

.text:00401640 ; void __thiscall TarArchive::extract(TarArchive *this)
.text:00401640 ?extract@TarArchive@@UAEXXZ proc near ; DATA XREF: .rdata:0041A388↓o
.text:00401640
.text:00401640 this          = dword ptr -4
.text:00401640
.text:00401640          push    ebp
.text:00401641          mov     ebp, esp
.text:00401643          push    ecx
.text:00401644          mov     [ebp+this], ecx
.text:00401647          mov     eax, ds:__imp__is_registered
.text:0040164C          movzx   ecx, byte ptr [eax]
.text:0040164F          push    ecx ; state
.text:00401650          call   ?set_tar_registration_state@@YAX_N@Z ; set_tar_registration_state(bool)
.text:00401655          add     esp, 4
.text:00401658          push    offset path ; path
.text:0040165D          mov     edx, [ebp+this]
.text:00401660          mov     eax, [edx+8]
.text:00401663          push    eax ; a
.text:00401664          call   ?untar@@YAXPAU_iobuf@@@PBD@Z ; untar(_iobuf *,char const *)
.text:00401669          add     esp, 8
.text:0040166C          mov     esp, ebp
.text:0040166E          pop     ebp
.text:0040166F          retn
.text:0040166F ?extract@TarArchive@@UAEXXZ endp

```

确定函数TarArchive::extract的地址 (Decompress.exe)

```

.text:100CE4D0 ; _iobuf *__cdecl fopen(const char *file, const char *mode)
.text:100CE4D0          public _fopen
.text:100CE4D0 _fopen          proc near ; DATA XREF: .text:10005E42↑o
.text:100CE4D0 ; __o_fopen+D↑o ...

.text:100CA670 ; int __cdecl fclose(_iobuf *public_stream)
.text:100CA670          public _fclose
.text:100CA670 _fclose        proc near ; CODE XREF: _anonymous_namespace___stream_traits__close+9↑p
.text:100CA670 ; __pclose+F2↑p ...

.text:100D25F0 ; unsigned int __cdecl fwrite(const void *buffer, unsigned int size, unsigned int count, _iobuf *stream)
.text:100D25F0          public _fwrite
.text:100D25F0 _fwrite        proc near ; DATA XREF: .text:10005EEC↑o
.text:100D25F0 ; __o_fwrite+15↑o ...

```

确定函数fopen/fclose/fwrite的地址 (ucrtbased.dll)

静态方式注入代码 (StaticCodeInjection)

```
1. void FakeArchive::ori_rar_extract(void) {}  
2. void FakeArchive::my_rar_extract(void) {  
3.     FILE *fp;  
  
4.     fp = fopen("README", "w+");  
5.     fwrite("Developed by UV.", 1, 16, fp);  
6.     fclose(fp);  
  
7.     ori_rar_extract();  
8. }  
  
9. int main() {  
10.     FakeArchive archive;  
11.     archive.my_rar_extract();  
12.     return 0;  
13. }
```

StaticCodeInjection可执行文件

Step1: 新增一个节.new, 改变.rdata节的大小
Step2: 在.new节注入代码, 在.rdata节注入数据
Step3: 对新注入的代码进行符号重定位
Step4: 更改虚函数表的表项

```
FakeArchive::my_rar_extract()  
{  
    /* my implementation */  
    RarArchive::extract();  
}
```

Decompress可执行文件

#	Name	Virtual Size	Virtual Offset	Raw Size	Raw Offset	Permission
1	.text	00018C94	00001000	00018E00	00000400	RE
2	.rdata	00008CB0	0001A000	00008E00	00019200	R
3	.data	000005C4	00023000	00000200	00022000	RW
4	.rsrc	000001F8	00024000	00000200	00022200	R
5	.new	00001000	00026000	00000200	00022800	RWE

Section
Table

```
RarArchive::extract()  
{/* ori implementation */}
```

.text

ptr to RarArchive::~RarArchive()

ptr to FakeArchive::my_rar_extract()

.rdata

```
FakeArchive::my_rar_extract()  
{  
    /* my implementation */  
    RarArchive::extract();  
}
```

.new

重定位本模块中定义的符号

重定位其它模块中定义的符号

; 可执行文件 (StaticCodeInjection)

```

.text:00401000 <-----
.text:0040100A C3
.text:0040100A
.text:00401010
.text:00401010
.text:00401010
.text:00401010 55
.text:00401011 8B EC
.text:00401013 83 EC 08
.text:00401016 89 4D F8
.text:00401019 68 08 31 40 00
.text:0040101E 68 0C 31 40 00
.text:00401023 FF 15 BC 30 40 00
.text:00401037 68 14 31 40 00
.text:0040103C FF 15 B8 30 40 00
.text:00401042 83 C4 10
.text:00401045 8B 4D FC
.text:00401048 51
.text:00401049 FF 15 C0 30 40 00
.text:0040104F 83 C4 04
.text:00401052 8B 4D F8
.text:00401055 E8 A6 FF FF FF
.text:0040105A 8B E5
.text:0040105C 5D
.text:0040105D C3
.text:0040105D

; void __thiscall FakeArchive::ori_rar_extract(FakeArchive *this)
;
; .....
```

```

.idata:004030B8 <-----, size_t __cdecl _fwrite(const void *Buffer, size_t ElementSize, size_t ElementCount, FILE *Stream)
.idata:004030BC <-----, FILE * __cdecl _fopen(const char *FileName, const char *Mode)
.idata:004030C0 <-----, int __cdecl _fclose(FILE *Stream)
```

```

.rdata:00403108 77 2B 00 Mode db 'w+',0
.rdata:0040310C 52 45 41 44 4D 45 00 FileName db 'README',0
.rdata:00403114 44 65 76 65 6C 6F 70 65+aDevelopedByUv db 'Developed by UV.',0 ;
```

```

.text:00401710 <-----
.text:0040173A C3
.text:0040173A

.new:00425000 00 00 00 00 00 00 00 00+ dword_425000 dd 4 dup(0) ; CODE XREF: .new:00425055,p
.new:00425010 ; -----
.new:00425010 55 push ebp
.new:00425011 8B EC mov ebp, esp
.new:00425013 83 EC 08 sub esp, 8
.new:00425016 89 4D F8 mov [ebp-8], ecx
.new:00425019 68 B0 2C 42 00 push 403108h
.new:0042501E 68 C0 2C 42 00 push offset loc_40310C
.new:00425023 FF 15 64 A1 41 00 call dword ptr ds:loc_4030BA+2

.new:00425035 6A 01 push 1
.new:00425037 68 D0 2C 42 00 push offset loc_403114
.new:0042503C FF 15 A8 A0 41 00 call dword ptr ds:loc_4030B7+1
.new:00425042 83 C4 10 add esp, 10h
.new:00425045 8B 4D FC mov ecx, [ebp-4]
.new:00425048 51 push ecx
.new:00425049 FF 15 68 A1 41 00 call dword ptr ds:loc_4030BF+1
.new:0042504F 83 C4 04 add esp, 4
.new:00425052 8B 4D F8 mov ecx, [ebp-8]
.new:00425055 E8 B6 C6 FD FF call near ptr dword_425000
.new:0042505A 8B E5 mov esp, ebp
.new:0042505C 5D pop ebp
.new:0042505D C3 retn

=====

.idata:0041A0A8 <-----, size_t __cdecl _fwrite(const void *Buffer, size_t ElementSize, size_t ElementCount, FILE *Stream)
.idata:0041A154 <-----, FILE * __cdecl _fopen(const char *FileName, const char *Mode)
.idata:0041A158 <-----, int __cdecl _fclose(FILE *Stream)

=====

.rdata:0041A390 ; void (__cdecl *const RarArchive::`vftable'[3])()
.rdata:0041A390 ; RarArchive::~~RarArchive(void)+A10
.rdata:0041A394 10 50 42 00 dd offset ?extract@RarArchive@@UAEXXZ ; RarArchive::extract(void)

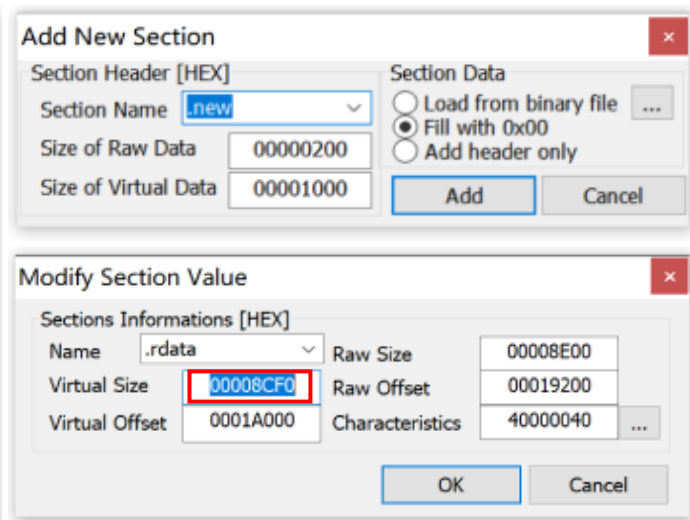
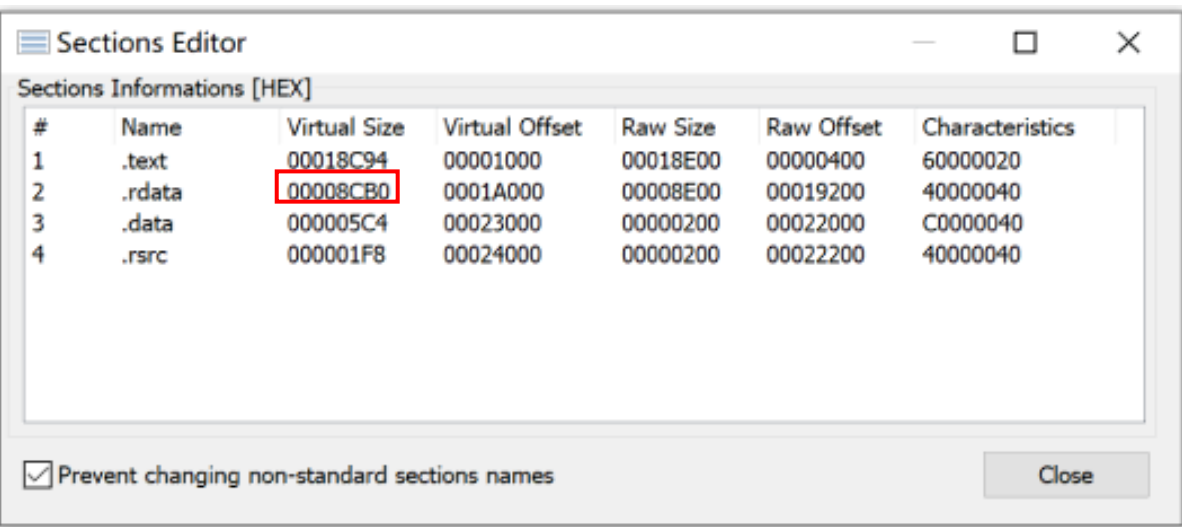
.rdata:00422C30 77 2B 00 aW db 'w+',0
.rdata:00422C30 52 45 41 44 4D 45 00 aReadme db 'README',0
.rdata:00422C30 44 65 76 65 6C 6F 70 65+aDevelopedByUv db 'Developed by UV.',0

```

```

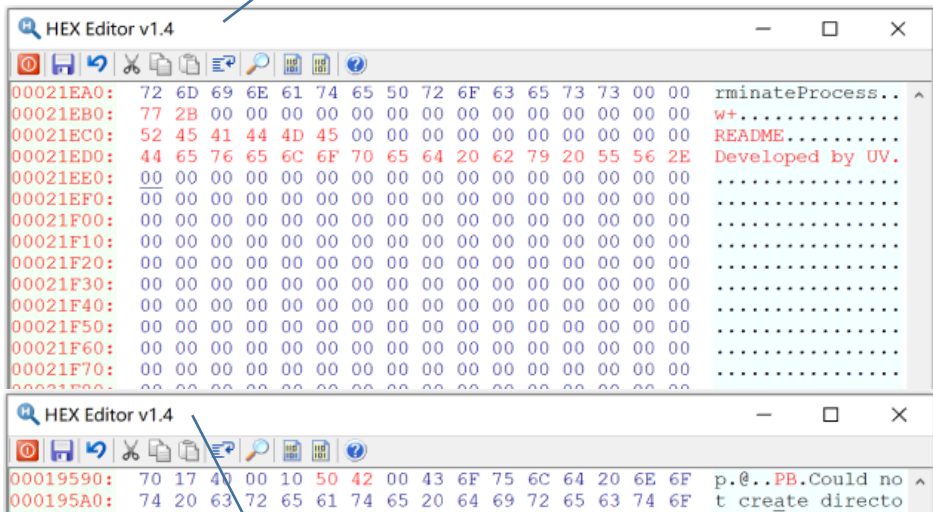
.text:000011ED ; printfloat(int)
.text:000011ED         public _Z10printfloati
.text:000011ED _Z10printfloati proc near                ; CODE XREF: main+2F1p
.text:000011ED
.text:000011ED var_28         = qword ptr -28h
.text:000011ED
.text:000011ED         rep nop ebx
.text:000011F1         push     ebp
.text:000011F2         mov      ebp, esp
.text:000011F4         push     ebx
.text:000011F5         sub      esp, 14h
.text:000011F8         call     __x86_get_pc_thunk_ax           ;用于处理PIC, 具体含义为将下一条指令的地址存入eax中
.text:000011FD         add      eax, 2DD7h
.text:00001202         fild     dword ptr [ebp+8]               ;num转换双精度数压入ST(0)
.text:00001205         fstp     dword ptr [ebp-0Ch]             ;f=(float) num
.text:00001208         fld      dword ptr [ebp-0Ch]             ;将f压入ST(0)中
.text:0000120B         sub      esp, 4
.text:0000120E         lea      esp, [esp-8]
.text:00001212         fstp     [esp+28h+var_28]
.text:00001215         lea      edx, (unk_2008 - 3FD4h)[eax]      ;参数2:f 入栈
.text:0000121B         push     edx                               ;获取格式串"%f"的地址
.text:0000121C         mov      ebx, eax                         ; 参数1:"%f" 入栈
.text:0000121E         call     sub_1080                         ;调用printf
.text:00001223         add      esp, 10h                        ; 平衡栈
.text:00001226         nop
.text:00001227         mov      ebx, [ebp-4]
.text:0000122A         leave
.text:0000122B         retn
.text:0000122B _Z10printfloati endp

```

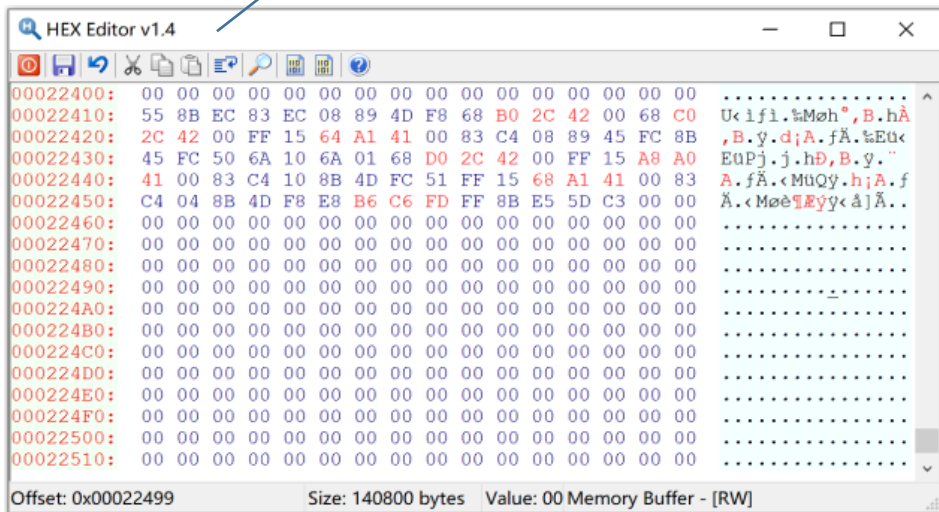


修改.rdata节, 添加关键字符串

修改.new节, 进行重定位



修改.rdata节, 更改虚函数表的表项



5 软件保护应用实例

- 防止分析
- 防止调试
- 防止修改

防止分析

观察黄色高亮处代码在插入花指令前后的反汇编结果

```
1. bool try_register(char* email, char* name, int code)
2. {
3.     ...
4.     __asm {
5.         JMP code_start0
6.         _EMIT 0x33
7.         code_start0:
8.     };
9.     if (computed_code == code) return true;
10.    else return false;
11. }
```

当反汇编器解析完jmp code_start0指令后，会接着从下一个0x33开始进行解析，而0x33为call指令的起始字节，就会导致反汇编器认为从0x33开始的5个字节为一条xor指令，从而让后续的若干条指令被错误解析。

插入的数据字节

插入的花指令

插入花指令前objdump结果:

200013b3:33 c2	xor	%edx,%eax
200013b5:03 45 f8	add	-0x8(%ebp),%eax
200013b8:89 45 f8	mov	%eax,-0x8(%ebp)
200013bb:eb d3	jmp	0x20001390
200013bd:8b 45 f8	mov	-0x8(%ebp),%eax
200013c0:3b 45 10	cmp	0x10(%ebp),%eax
200013c3:75 06	jne	0x200013cb
200013c5:b0 01	mov	\$0x1,%al
200013c7:eb 04	jmp	0x200013cd
200013c9:eb 02	jmp	0x200013cd
200013cb:32 c0	xor	%al,%al
200013cd:8b e5	mov	%ebp,%esp
200013cf:5a	pop	%ebp
200013d0:c3	ret	

正确的反汇编代码

插入花指令后objdump结果:

200013b6:33 c2	xor	%edx,%eax
200013b8:03 45 f8	add	-0x8(%ebp),%eax
200013bb:89 45 f8	mov	%eax,-0x8(%ebp)
200013be:eb d3	jmp	0x20001393
200013c0:eb 01	jmp	0x200013c3
200013c2:33 8b 45 f8 3b 45	xor	0x453bf845(%ebx),%ecx
200013c8:10 75 06	adc	%dh,0x6(%ebp)
200013cb:b0 01	mov	\$0x1,%al
200013cd:eb 04	jmp	0x200013d3
200013cf:eb 02	jmp	0x200013d3
200013d1:32 c0	xor	%al,%al
200013d3:5f	pop	%edi
200013d4:5e	pop	%esi
200013d5:5b	pop	%ebx
200013d6:8b e5	mov	%ebp,%esp
200013d8:5d	pop	%ebp
200013d9:c3	ret	

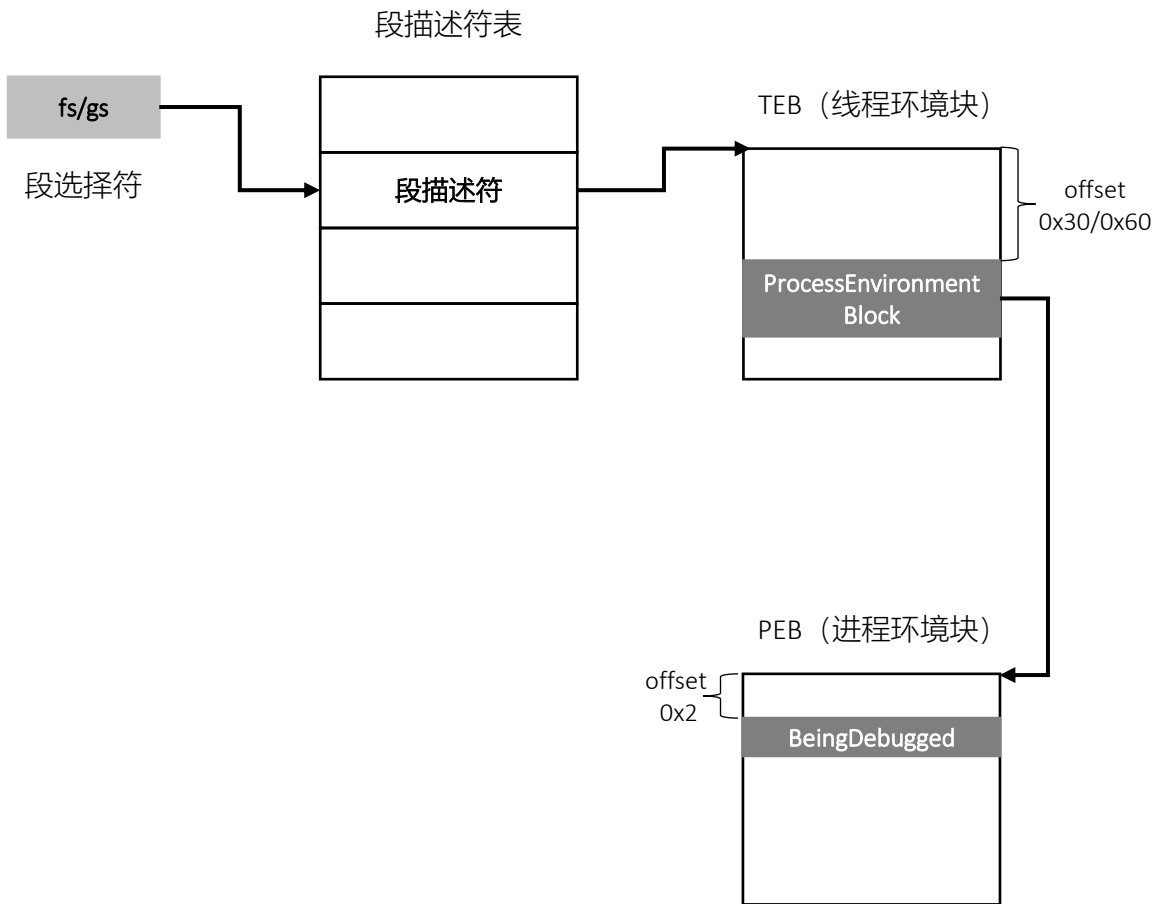
错误的反汇编代码

插入花指令后，try_register函数返回值计算逻辑难以分析!

防止调试

```
1.  int IsDebuggerPresent(void)
2.  {
3.      #ifdef _WIN64
4.          __asm
5.          {
6.              mov rax, dword ptr gs : [0x60]
7.              movzx rax, byte ptr ds : [rax + 0x2]
8.          }
9.      #else
10.         __asm
11.         {
12.             mov eax, dword ptr fs : [0x30]
13.             movzx eax, byte ptr ds : [eax + 0x2]
14.         }
15.     #endif
16. }

17. void anti_debugging(void)
18. {
19.     if (IsDebuggerPresent())
20.     {
21.         printf("Err: Under debugging.\n");
22.         exit(1);
23.     }
24.     return 0;
25. }
```



防止修改

```
1. void anti_modification(void)
2. {
3.     DWORD PEHeaderChecksum, PEComputedChecksum;
4.     DWORD MemComputedChecksum;
5.     HMODULE hMod;
6.     IMAGE_DOS_HEADER* pDosHeader;
7.     IMAGE_NT_HEADERS* pNTHHeader;
8.     IMAGE_FILE_HEADER* pFileHeader;
9.     IMAGE_OPTIONAL_HEADER* pOptHeader;
10.    IMAGE_SECTION_HEADER* pSecHeader;
11.    char path[MAXLEN];
12.    int i; byte *b;

13.    PEHeaderChecksum = 0, PEComputedChecksum = 0;

14.    GetModuleFileName(NULL, path, MAXLEN);
15.    MapFileAndCheckSum(path, &PEHeaderChecksum,
16.                        &PEComputedChecksum);

17.    if (PEHeaderChecksum != PEComputedChecksum)
18.    {
19.        printf("Err: PE file has been modified. %x vs %x\n",
20.              PEHeaderChecksum, PEComputedChecksum);
21.        exit(1);
22.    }

23.    hMod = GetModuleHandle(NULL);
24.    pDosHeader = (IMAGE_DOS_HEADER*)(hMod);
25.    pNTHHeader = (IMAGE_NT_HEADERS*)((BYTE*)(hMod) +
26.                                         pDosHeader->e_lfanew);
27.    pFileHeader = (IMAGE_FILE_HEADER*)((BYTE*)(hMod) +
28.                                         pDosHeader->e_lfanew + 4);
```

```
29.    pOptHeader = (IMAGE_OPTIONAL_HEADER*)((BYTE*)
30.                                             (pNTHHeader) + sizeof(pNTHHeader->Signature) +
31.                                             sizeof(pNTHHeader->FileHeader));
32.    pSecHeader = (IMAGE_SECTION_HEADER*)((LPBYTE)(pOptHeader)
33.                                           + pFileHeader->SizeOfOptionalHeader);

34.    for (i = 0; i < pFileHeader->NumberOfSections; i++)
35.    {
36.        if (!strcmp((const char*)(pSecHeader->Name), ".rdata"))
37.        {
38.            MemComputedChecksum = 0;

39.            for (b = (byte*)(hMod) + pSecHeader->VirtualAddress;
40.                 b < (byte*)(hMod) + pSecHeader->VirtualAddress +
41.                     pSecHeader->Misc.VirtualSize; b++)
42.            {
43.                MemComputedChecksum += *b;
44.            }

45.            if (SavedMemComputedChecksum &&
46.                MemComputedChecksum != SavedMemComputedChecksum)
47.            {
48.                printf("Err: memory has been modified. %x vs %x\n",
49.                      SavedMemComputedChecksum, MemComputedChecksum);
50.                exit(1);
51.            }

52.            SavedMemComputedChecksum = MemComputedChecksum;
53.        }
54.        pSecHeader++;
55.    }
56. }
```

作业

- 理解本章提供的示例代码
- 编写脚本，实现“差分分析内存读写，确定关键计数变量的地址”
- 使用动态代码注入技术，拦截RarArchive::extract()函数；
使用静态代码注入技术，拦截TarArchive::extract()函数；
- 尝试绕开“防止调试”实例和“防止修改”实例