

题型

两个计算分析题 20分 10/个 计算结果 过程

7个简答题 56 8分 /个

两个综合题 两个

简答题 基本送分 比例大

计算分析题这门课设计不多 有些做分析

综合题 考到漏洞

答题不允许上网

三部分 基础部分 第二讲

pe文件结构出计算题 分析题 可能性大（必考 抽得到）

资源 引入机制（重点） 没有实操（棒棒）重定位 整体难度降了不少

考试拓展不多，还好

综合题四小问

指南覆盖很多 简答题

磁盘肯定有题

处理器工作模式了解了解就可以

内存布局分配 需要多了解 dep使用

计算机引导 大概了解

PE文件格式 重点了解

漏洞利用 cvss计算不考

软件漏洞分类 缺陷 怎么产生

攻防过程要了解 攻防博弈

缓冲区溢出 堆 web还是要了解 跨站攻击 整形溢出 格式化字符串熟悉下 //说不定遇到

ppt栈帧 画缓冲区堆栈结构 看ppt

[illegible]

- 2) 下图为某 PE 程序的节表信息及引入表 (RVA: 0x15000) 的信息。请问:
- 该程序从哪几个 dll 分别引入了哪些函数?
 - 当该程序装载到内存后, 上述被引入函数的真实内存地址分别将被填充到内存的什么位置 (RVA) ?

C:\documents and settings\administrator\桌面\backup\演示\nc.exe

文件头 | Dos | 区段 | 函数 | 资源 | 签名 | 进程 | 选项

| No | 名称 | 虚拟大小 | 虚拟偏移 | 实际大小 | 实际偏移 | 特征码 |
|----|------|----------|----------|----------|----------|----------|
| 01 | UPX0 | 00000000 | 00001000 | 00000000 | 00000200 | E0000080 |
| 02 | UPX1 | 00007000 | 0000E000 | 00006A00 | 00000200 | E0000040 |
| 03 | UPX2 | 00001000 | 00015000 | 00000200 | 00006C00 | C0000040 |

UltraEdit - [C:\Documents and Settings\Administrator\桌面\backup\演示\nc.exe]

文件(F) 编辑(E) 搜索(S) 方案(P) 视图(V) 格式(T) 列(L) 宏(M) 脚本(S) 高级(A) 窗口(W) 帮助(H)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | |
|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------------------|
| 00006bc0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; |
| 00006bd0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; |
| 00006be0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; |
| 00006bf0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; |
| 00006c00h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 54 | 50 | 01 | 00 | ;TP.. |
| 00006c10h: | 3C | 50 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; <P..... |
| 00006c20h: | 61 | 50 | 01 | 00 | 4C | 50 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; aP..LP..... |
| 00006c30h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6C | 50 | 01 | 00 | ;lP.. |
| 00006c40h: | 7A | 50 | 01 | 00 | 8A | 50 | 01 | 00 | 00 | 00 | 00 | 00 | 13 | 00 | 00 | 80 | ; zP..着.....e |
| 00006c50h: | 00 | 00 | 00 | 00 | 4B | 45 | 52 | 4E | 45 | 4C | 33 | 32 | 2E | 44 | 4C | 4C | ; ...KERNEL32.DLL |
| 00006c60h: | 00 | 57 | 53 | 4F | 43 | 4B | 33 | 32 | 2E | 64 | 6C | 6C | 00 | 00 | 4C | 6F | ; .WSOCK32.dll..Lo |
| 00006c70h: | 61 | 64 | 4C | 69 | 62 | 72 | 61 | 72 | 79 | 41 | 00 | 00 | 47 | 65 | 74 | 50 | ; adLibraryA..GetP |
| 00006c80h: | 72 | 6F | 63 | 41 | 64 | 64 | 72 | 65 | 73 | 73 | 00 | 00 | 45 | 78 | 69 | 74 | ; rocAddress..Exit |
| 00006c90h: | 50 | 72 | 6F | 63 | 65 | 73 | 73 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; Process..... |
| 00006ca0h: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ; |

2. 简答题 (6*8=48 分)

- 关于计算机信息系统犯罪, 根据我国刑法规定, 主要涉及到哪几条罪? 如何量刑?
- 为什么需要进行地址重定位? 正常 PE 程序以及恶意代码在重定位上有何差异?
- 熊猫烧香病毒在感染目标文件后, 目标程序图标将变成熊猫图案, 请问其原因是什么? 对于熊猫烧香病毒的感染方式而言, 怎样才能够做到被感染程序的图标不发生变化?
- 什么是 DWORD SHOOT, 请简要描述其机理。
- 漏洞的通用阻断技术包括 GS、DEP、ASLR、SafeSEH 等, 请各自通过一句话简要描述他们的作用和机理?

- 以下为 EXP 中 payload 的常见结构, 请对该段数据各个字段的含义进行解释, 如果删除 `make_nops(20)` 这部分 `nop` 指令, 将可能导致什么结果?

3. 综合题 (10+12+14=36 分)

(1) (10 分) 下面是在 Linux 下用 C 语言编写的一个有漏洞的函数，左边是源代码，右边是编译后的汇编代码：

| | |
|--|---|
| <pre>#include <unistd.h> void foo() { char buffer[128]; read(0, buffer, 200); }</pre> | <pre>0804843B push ebp 0804843C mov ebp, esp 0804843E sub esp, 88h 08048444 sub esp, 4 08048447 push 0C8h 0804844C lea eax, [ebp-88h] 08048452 push eax 08048453 push 0 08048455 call _read 0804845A add esp, 10h 0804845D nop 0804845E leave 0804845F retn</pre> |
|--|---|

假设 ASLR 已关闭，libc.so 的加载基址为 0xf7e07000，system 函数在 libc 中的偏移为 0x3A940，在 libc.so 中找到一个“/bin/sh”字符串，偏移为 0x15902b，回答下列问题：

- 该函数存在什么漏洞？buffer 开始位置距离该函数的返回地址有多少个字节？system 函数和“/bin/sh”字符串在内存中的虚拟地址分别为多少？
- 请编写一段 payload，使得在执行完 foo 函数后能够执行 system(“/bin/sh”)。

(2) (12 分) 以下是通过 mona 构建的一段 ROP 链，请分析并回答下列问题：

- 该段 ROP 获得控制权之后执行的第一条指令的地址是什么？
- 请画出最后一行地址 (0x77e5d13f) 对应的代码 PUSHAD 运行之后的堆栈布局，此时各个寄存器的值及其含义是什么？
(提示：pushad 依次将 EAX,ECX,EDX,EBX,ESP,EBP,ESI 和 EDI 压入堆栈。)
- 最后一行地址 (0x77e5d13f) 对应的代码 RETN 运行之后，将发生什么？

```
# rop chain generated with mona.py - www.corelancore.com
rop_gadgets =
[
    0x77c04fcd, # POP EAX # RETN [msvcrt.dll]
    0x77da1404, # ptr to &SetInformationProcess() [IAT ADVAPI32.dll]
    0x77e693d4, # MOV EAX,DWORD PTR DS:[EAX] # RETN [RPCRT4.dll]
    0x77e8f419, # XCHG EAX,EBP # RETN [RPCRT4.dll]
    0x77e5d13f, # RETN [RPCRT4.dll]
```


(3) (14 分) 现有下面一段用来进行身份认证的代码，左边是源代码，右边是 **Login** 函数的反汇编代码片段。这是一段认证程序，但是看起来是不可能通过认证的，因为认证使用的密码是每次随机生成的。然而，这段程序中有一些漏洞，使得我们有机会通过认证。

| | |
|--------------------------------------|---------------------------------|
| #include <stdio.h> | 080485D4 push ebp |
| #include <stdlib.h> | 080485D5 mov ebp, esp |
| #include <unistd.h> | 080485D7 sub esp, 118h |
| | 080485DD mov eax, large gs:14h |
| #define BUF_SIZE 128 | 080485E3 mov [ebp-0Ch], eax |
| | 080485E6 xor eax, eax |
| void LoginSuccess() | 080485E8 call _rand |
| { | 080485ED mov [ebp-114h], eax |
| printf("Welcome!\n"); | 080485F3 sub esp, 0Ch |
| system("/bin/sh"); | 080485F6 push offset aEnterName |
| } | 080485FB call _puts |
| | 08048600 add esp, 10h |
| void Login() | 08048603 sub esp, 4 |
| { | 08048606 push 0C8h |
| int randInt = rand(); | 0804860B lea eax, [ebp-10Ch] |
| | 08048611 push eax ; buf |
| char name[BUF_SIZE]; | 08048612 push 0 ; fd |
| printf("Enter your name:\n"); | 08048614 call _read |
| read(0, name, 200); | 08048619 add esp, 10h |
| | 0804861C sub esp, 0Ch |
| char pass[BUF_SIZE]; | 0804861F lea eax, [ebp-10Ch] |
| printf(name); | 08048625 push eax |
| printf(" please enter password:\n"); | 08048626 call _printf |

| | |
|--------------------------------------|-----------------------------------|
| printf("Enter your name:\n"); | 08048614 call _read |
| read(0, name, 200); | 08048619 add esp, 10h |
| | 0804861C sub esp, 0Ch |
| char pass[BUF_SIZE]; | 0804861F lea eax, [ebp-10Ch] |
| printf(name); | 08048625 push eax |
| printf(" please enter password:\n"); | 08048626 call _printf |
| read(0, pass, 200); | 0804862B add esp, 10h |
| | 0804862E sub esp, 0Ch |
| int passInt = atoi(pass); | 08048631 push offset aPleaseEnter |
| if(passInt == randInt) | 08048636 call _puts |
| LoginSuccess(); | 0804863B add esp, 10h |
| else | 0804863E sub esp, 4 I |
| printf("Wrong password.\n"); | 08048641 push 0C8h |
| } | 08048646 lea eax, [ebp-8Ch] |
| | 0804864C push eax ; buf |
| int main(int argc, char *argv[]) | 0804864D push 0 ; fd |
| { | 0804864F call _read |
| srand(time(NULL)); | |
| Login(); | 08048695 mov eax, [ebp-0Ch] |
| return 0; | 08048698 xor eax, large gs:14h |
| } | 0804869F jz short locret_80486A6 |
| | 080486A1 call ____stack_chk_fail |
| | 080486A6 |
| | 080486A6 locret_80486A6: |
| | 080486A6 leave |
| | 080486A7 retn |

(提示：在 Linux 中，`printf` 函数使用 `$` 符能够选择打印指定位置的可选参数，例如 `printf("%2$08x", 0x0000000b, 0x0000000a)` 将打印出第二个参数，即“0000000a”)。

`LoginSuccess` 函数的地址为 0x080485AB。

请回答下列问题：

- (1) 程序中包含哪几种类型的漏洞？
- (2) 想要通过认证，调用 `LoginSuccess`，有哪些思路？
- (3) 请选择其中一条思路，给出两次输入的 payload 以通过认证，并对 payload 内容进行解释 (包括各填充数据大小以及偏移的计算方法，泄露数据用 `leak data` 表示)。

✓ 平时分 (50%)

✧ 珞珈在线平台学习 (包含日常学习、作业、测试等)
【30%】

✧ 分组大作业【每组最多4人】【20%】

✧ 课后加分题

✧ 技术题【数量：10+】

✧ 前三位完成的同学给与5-10%的奖励。

✧ 用于补充平时分，分满为止。

✧ 其他 (加分5%)

✧ 论坛讨论、解答、部分实践环节的视频制作等

✓ 期末考试 (开卷, 50%)