2020-2021 学年度第 一 学期
《软件安全》期末考试试卷 A 卷（开 卷）【参考】

专业：_____ **学号：**_____ **姓名：**_____

说明：答案请全部写在答题纸上，写在试卷上无效。
未经主考教师同意，考试试卷、答题纸、草稿纸均不得带离考场，否则视为违规。

题号	一	二	三				总分
分值	20	56	24				100

一. 计算与分析题（共 2 小题，每小题 10 分，共 20 分）

1. 下是某 MBR 分区格式硬盘的分区表信息，请问：
   （1） 该磁盘包含哪几种类型的分区？请说明理由。（4 分）
   （2） 请给出各分区起始和结束扇区位置，以及分区的大小（给出计算公式即可）。（6 分）

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000001B0	00	00	00	00	00	2C	44	63	6E	D0	6E	D0	00	00	80	01
0000001C0	01	00	07	FE	FF	FF	3F	00	00	00	00	00	C0	03	00	FE
0000001D0	FF	FF	0F	FE	FF	FF	43	00	C0	03	00	00	80	02	00	00
0000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AA

某 MBR 分区格式硬盘的分区表信息

2. 以下是一个 C 语言编写的代码片段，要求：
   （1） 分析其中存在的安全缺陷（3 分）
   （2） 说明上述缺陷可能带来的安全风险（3 分）
   （3） 给出上述缺陷的一种修复方案（4 分）

行号	代码
（1）	char *pKeyTab;
（2）	char * genKeyString() {
（3）	char *pKey = NULL;
（4）	// 生成并返回一个字符串 pKey，代码略
（5）	// ……
（6）	return pKey;
（7）	}
（8）	
（9）	void initKeys(int number) {
（10）	if (number > 0) {
（11）	pKeyTab = malloc(number *sizeof(char*));
（12）	for (i = 0; i < number; i++)
（13）	pKeyTab[i] = genKeyString();
（14）	}
（15）	}

二. 简答题（共 7 小题，每小题 8 分，共 56 分）

1. 近年来国家在网络空间安全领域的立法日臻完善，通过本课程的学习，谈谈你对网络安全从业人员合法运用专业技能的必要性的认识。

2. 通过恶意代码样本，可以进行黑客个人及组织的溯源工作。请从 PE 文件静态分析角度，给出不少于四类恶意代码溯源特征，并说明溯源依据。

3. 根据你对 PE 格式的理解，给出在确保 PE 文件正常功能的前提下，缩减其大小的可行思路。

4. 如果怀疑自己电脑被远程木马控制，应当如何进行检测？请给出你的思路。

5. GS 技术可以有效阻止部分漏洞攻击，请简要描述其阻止机理，并给出绕过方法。

6. Shellcode 与 PE 感染病毒代码一样，都需要获取 API 函数地址。请问，与 PE 病毒感染代码相比，其存在哪些特殊要求，如何解决？

7. 某同学在做基于 ROP 的 DEP 绕过实验时，用 mona 工具搜索到如下所示的 ROP 链，但是在利用时总是无法按预期正确调用到 VirtualProtect 函数。通过调试，发现当 EIP=0x00b8c354 时，[ESP]=0x00b23eea。请分析导致这个问题的原因，并说明该如何修正下述 ROP 链，使得 VirtualProtect 可被顺利调用。

ROP 链
0x00b8c354, // POP EAX // RETN 0x00f00688, // ptr to &VirtualProtect() [IAT exp04-2.exe] 0x008bd826, // MOV EAX,DWORD PTR DS:[EAX] // RETN 0x00b23eea, // XCHG EAX,ESI // RETN 0x008b7206, // POP EBP // RETN 0x008f8d62, // & jmp esp 0x00dd9005, // POP EBX // RETN 0x00000201, // 0x00000201-> ebx 0x00b29075, // POP EDX // RETN 0x00000040, // 0x00000040-> edx 0x00b1dfb6, // POP ECX // RETN 0x00ef2be9, // &Writable location 0x00da2731, // POP EDI // RETN 0x00b13e04, // RETN (ROP NOP) 0x00b12f47, // POP EAX // RETN 0x90909090, 0x00b250d2, // PUSHAD // RETN

三. 综合题（共 2 小题，每小题 12 分，共 24 分）

1. 下图为 Windows 下某 PE 文件的片段截图，请问：

（1） 程序的引入函数目录表（IDT）表开始位置在该 PE 文件的文件偏移位置是多少？（4 分）

（2） IDT 表起始位置对应的默认 VA 地址是多少？（2 分）

（3） 该程序从多少个 dll 中引入了 API 函数？（3 分）

（4） 该程序从所有 dll 引入的总 API 函数个数为多少？（3 分）

```
0100h: 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00  ........PE..L...
0110h: EF 17 F0 5D 00 00 00 00 00 00 00 00 E0 00 02 01  . ........ ...
0120h: 0B 01 0E 10 00 54 05 00 00 CC 0A 00 00 00 00 00  .....T... ......
0130h: BF C3 00 00 00 10 00 00 00 70 05 00 00 00 40 00  . .......p....@.
0140h: 00 10 00 00 00 02 00 00 05 00 01 00 00 00 00 00  ................
0150h: 05 00 01 00 00 00 00 00 00 60 10 00 00 04 00 00  .........`......
0160h: 25 38 10 00 02 00 40 81 00 00 10 00 00 10 00 00  %8....@.........
0170h: 00 00 10 00 00 10 00 00 00 00 00 00 10 00 00 00  ................
0180h: 00 00 00 00 00 00 00 00 8C 6B 06 00 F0 00 00 00  .........k.. ...
0190h: 00 C0 06 00 90 50 09 00 00 00 00 00 00 00 00 00  . ...P..........
01A0h: 00 0A 10 00 78 23 00 00 00 20 10 00 B0 39 00 00  ....x#... ...9..
01B0h: A0 58 06 00 54 00 00 00 00 00 00 00 00 00 00 00  .X..T...........
01C0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01D0h: F8 58 06 00 40 00 00 00 00 00 00 00 00 00 00 00  .X..@...........
01E0h: 00 70 05 00 40 04 00 00 00 00 00 00 00 00 00 00  .p..@...........
01F0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0200h: 2E 74 65 78 74 00 00 00 ED 52 05 00 00 10 00 00  .text... .......
0210h: 00 54 05 00 00 04 00 00 00 00 00 00 00 00 00 00  .T..............
0220h: 00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00  .... ..`.rdata..
0230h: 40 13 01 00 00 70 05 00 00 14 01 00 00 58 05 00  @....p.......X..
0240h: 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
0250h: 2E 64 61 74 61 00 00 00 24 2A 00 00 00 90 06 00  .data...$*......
0260h: 00 12 00 00 00 6C 06 00 00 00 00 00 00 00 00 00  .....l..........
0270h: 00 00 00 00 40 00 00 C0 2E 72 73 72 63 00 00 00  ....@.. .rsrc...
0280h: 90 50 09 00 00 C0 06 00 00 52 09 00 00 7E 06 00  .P... ...R...~..
0290h: 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40  ............@..@
02A0h: 2E 72 65 6C 6F 63 00 00 B0 39 00 00 00 20 10 00  .reloc...9... ..
02B0h: 00 3A 00 00 00 D0 0F 00 00 00 00 00 00 00 00 00  .:... ..........
02C0h: 00 00 00 00 40 00 00 42 00 00 00 00 00 00 00 00  ....@..B........
```

Windows 下某 PE 文件的片段截图

2. 下表列出了一个猜数字游戏的部分代码，该游戏在连续猜对 10 次后才可获得奖励，试对代码进行分析并回答下列问题：

（1） 结合表中代码，给出 main 局部变量的位置并画出栈布局(不考虑传入参数)；（4 分）

（2） 结合栈布局分析代码存在的缺陷；（4 分）

（3） 给出利用该缺陷获取奖励的思路。（4 分）

部分源代码 & 汇编代码

```
1.  int main(int argc, char **argv)
2.  {
3.    int guess_num;
4.    int i;
5.    int random_num;
6.    unsigned int seed;
7.    char name[10];
8.
9.    guess_num = 0;
10.   random_num = 0;
11.
12.   seed = get_seed();
13.   puts("This is a guess number game!");
14.   puts("Please input your name!");
15.   printf("Your name:");
16.   gets(name);
17.   srand(seed);
18.   for ( i = 0; i <= 9; ++i )
19.   {
20.     random_num = rand() % 9 + 1;
21.     printf("---------Turn:%d---------\n", i + 1);
22.     printf("Please input your guess number:");
23.     scanf("%d", &guess_num);
24.     if ( guess_num != random_num )
25.     {
26.       puts("The number is Wrong and Game Over!");
27.       exit(1);
28.     }
29.     puts("Right!");
30.   }
31.   get_award();
32.   return 0;
33. }
```

```
text:004010C0 <main>:
10C0    push    ebp
10C1    mov     ebp, esp
10C3    sub     esp, 5Ch
10C6    push    ebx
10C7    push    esi
10C8    push    edi
10C9    lea     edi, [ebp-5Ch]
10CC    mov     ecx, 17h
10D1    mov     eax, 0CCCCCCCCh
10D6    rep stosd
10D8    mov     dword ptr [ebp-4], 0
10DF    mov     dword ptr [ebp-0Ch], 0
10E6    call    j__get_seed
10EB    mov     [ebp-10h], eax
10EE    push    4270CCh
10F3    call    _puts
10F8    add     esp, 4
10FB    push    4281C4h
1100    call    _puts
1105    add     esp, 4
1108    push    4270C0h
110D    call    _printf
1112    add     esp, 4
1115    lea     eax, [ebp-1Ch]
1118    push    eax          ; buffer
1119    call    _gets
111E    add     esp, 4
1121    mov     ecx, [ebp-10h]
1124    push    ecx              ; seed
1125    call    _srand
112A    add     esp, 4
112D    mov     dword ptr [ebp-8], 0
……
```

附注：

（1） malloc 函数原型是： void* malloc(unsigned int size)；

（2） VirtualProtect 的函数原型是：BOOL VirtualProtect(LPVOID lpAddress, DWORD dwSize, DWORD flNewProtect, PDWORD lpflOldProtect)；

（3） gets 从 stdin 流中读取字符串，直至接受到换行符或 EOF 时停止，并将读取的结果存放在 buffer 指针所指向的字符数组中；

（4） srand 函数是随机数发生器的初始化函数，原型是：void srand(unsigned int seed)；srand 和 rand()配合使用产生伪随机数序列。