


《软件安全》期末考试试题 (A)

考 试 注 意 事 项	一、学生参加考试须带学生证或学院证明，未带者不准进入考场。学生必须按照监考教师指定座位就坐。 二、书本、参考资料、书包等物品一律放到考场指定位置。 三、学生不得另行携带、使用稿纸，要遵守《北京邮电大学考场规则》，有考场违纪或作弊行为者，按相应规定严肃处理。 四、学生必须将答题内容做在试题卷上，做在试题及草稿纸上一律无效。 五、学生的姓名、班级、学号、班内序号等信息由教材中心统一印制。
考 试 课 程	考试时间
题 号	一 二 三 四 五 六 七 八 总分
编 分	
得 分	
阅 卷 教 师	

一、 填空 (10 分，每空 0.5 分，每题各空无顺序关系)

- 函数栈帧是系统实现函数调用过程需要维护的基本数据结构，函数栈帧的基本信息包括：_____、_____和_____。
- 操作系统中进程所使用的内存根据用途不同，按照功能主要可分成_____、_____和_____四大部分。
- 常见的字符串操作错误主要包括：_____、_____、_____等。
- Fuzz 测试的主要目的包括：_____、_____和_____。
- 采用补码表示，一个 8 比特位无符号整数可以表示的范围是 0 ~ 255，一个 8 比特位有符号整数可以表示的范围是 -128 ~ 127。

二、 单项选择填空题 (30 分，每空 2 分)

- 下列无符号类型装换中，会造成数据曲解的是 (C)
 - A. unsigned char short
 - B. unsigned char long
 - C. unsigned short short
 - D. unsigned short long
- 下已对于如下代码，为了使 foo 中覆写的地址为 0x10402010，□处应该填入

- ```

 printf ("%16u%n%16u%n%32u%n%□u%n",
 1, (int *) &foo[0], 1, (int *) &foo[1],
 1, (int *) &foo[2], 1, (int *) &foo[3]);

```
- (A) 224 (B) 208 (C) 64 (D) 128
  - 关于标准 C 语言字符串，以下表述中，不正确的是 (D)。
    - (A) 由一个连续的字符序列组成，并以一个空字符 (NULL) 作为结束；
    - (B) 一个指向字符串的指针实际上就是指向该字符串起始字符；
    - (C) 字符串长度是指字符串实际所占存储空间的字节数，含空字符、不含空字符之后的区域；
    - (D) 字符串的值是指它所包含的按顺序排列的字符序列。
  - gets(), puts(), strcpy(), strcat(), strlen(), printf(), memcpy() 和 memset() 等 8 个 API 均来自标准 C，其中能够导致无界字符串越界写操作的有 ( ) 个。
    - (A) 3; (B) 4; (C) 5; (D) 6。
  - 下列 Windows 机制中 不是针对缓冲区溢出的技术为：(B)。
    - A. GS 编译技术 B. DEP
    - C. ASLR D. S.E.H
  - 函数调用过程中，指向栈顶和栈底的指针会发生变化，如图的指令组合通常发生在 (B)。
    - 1. push ebp
    - 2. mov ebp, esp
    - 3. sub esp, 28
    - ...

- 函数调用过程：(B) 被调函数的头开部分；(C) 被调函数的结尾部分；(D) 函数中控制流执行过程。
- 对于如下代码的执行结果：
 

```

char c;
printf ("%100u%n", 1, &c);

```

 c 的值用十六进制表示为 ( )。
  - A. 01 B. 64 C. 100 D. 04
- 缓冲区溢出能被用于覆写函数或数据指针，必须 (A) 同时成立。
  - (A) (1)、(2)、(4)；(B) (1)、(2)、(5)；(C) (1)、(2)、(3)、(4)；(D) (1)、(2)、(3)、(5)。
  - (1) 缓冲区与目标函数或者数据指针必须分配在同一个段内
  - (2) 缓冲区必须可以被缓冲区溢出利用
  - (3) 缓冲区的数据类型必须与目标指针的类型相同
  - (4) 缓冲区必须位于比目标函数或者数据指针更低的地址处
  - (5) 缓冲区必须位于比目标函数或者数据指针更高的地址处
- 函数栈帧是函数调用过程中使用的重要数据结构，函数栈帧包含的数据内容 (B)。
  - (A) 函数名
  - (B) 函数名、参数、局部变量、返回地址
  - (C) 函数名、参数、局部变量
  - (D) 函数名、参数、返回地址

(A) (1), (2); (B) (1), (2), (3); (C) (1), (2), (3), (4); (D) (1),

(2), (3), (4), (5);

(1) 调用者栈帧信息;

(2) 函数参数信息;

(3) 函数局部变量;

(4) 函数静态变量;

(5) 函数全局变量;

15. 在下列程序可以被用来实现任意内存写, 则以下说法错误的是 (D),

```
1. void foo(void * arg, size_t len) {
```

```
2. char buff[100];
```

```
3. long val = ...;
```

```
4. long *ptr = ...;
```

```
5. memcpy(buff, arg, len);
```

```
6. *ptr = val;
```

```
7. ...
```

```
8. return;
```

```
9. }
```

A. 第 3 行在溢出缓冲区后, 攻击者可以覆写 ptr 和 val

B. 第 4 行的指针接下来被用作一个赋值操作的目的地址

C. 第 5 行是有界复制, 不会导致溢出

D. 第 6 行会发生任意内存写

16. 运行 `printf("%s%s%s%s%s%s%s%s%s%s%s%s%s%s");` 会产生什么情况 (B)

A. 输出 `printf` 栈帧处的参数字符串

B. 触发无效指针存取或未映射的地址读取

C. 输出从 `printf` 栈底开始的数个字符串

D. 不输出任何内容

17. 以下程序预期的打印结果有 (C) 种情况。

(A) 1; (B) 2; (C) 3; (D) 不确定。

```
1. int main(int argc, char* argv[]) {
2. char source[10];
3. strcpy(source, "0123456789");
4. char *dest = (char *)malloc(strlen(source));
5. for (int i=1; i <= 11; i++) {
6. dest[i] = source[i];
7. }
8. dest[i] = '\0';
9. printf("dest = %s", dest);
10. }
```

18. 以下程序预期的输出结果是 (A)

```
signed char c1, c2, c;
```

```
c1 = 120;
```

```
c2 = 120;
```

```
c = c1 + c2;
```

```
printf("%d", c);
```

A. 240 B. -16

C. -15

D. -113

19. 为了防止整数的溢出, 一般需要进行先验条件测试, 以下对于先验条件的叙述错误的是 (D)

A. 将两个操作数放到下一个更大的数据类型上, 然后相乘。

B. 对于无符号整数检查下一个大整数的高阶位, 如果被设置了, 抛出错误。

C. 对带符号整数, 如果结果的高半部分及低半部分的符号位全为 0 或 1, 则没有发生整数溢出。

D. 为了防止无符号整数相乘时发生溢出, 检验 `A * B >= MAX_INT` 是否成立。

20. 下列关于地址的随机化, 有误的一项是 (C)

A. 地址的随机化可以使 `malloc` 等函数每次返回的地址是随机的, 这让程序每次展现出不同的地址空间行为。

B. 随机化会让程序的调试工作更加困难。

C. 地址随机化可以带来更小的性能开销和更好的运行效果。

D. 地址随机化使得漏洞利用更加困难。

### 三、简答题 (36 分, 每题 6 分)

21. 对比植入 shellcode 中静态淹没返回地址与 JMP ESP 两种方式的优缺点。

22. 请解释为什么格式化输出可能导致程序崩溃?

23. 简述软件漏洞能够导致的后果有哪些?

24. 指出并分析下图程序的整数溢出问题。

```
1. void getComment(unsigned int len, char *src) {
2. unsigned int size;
3. size = len - 2;
4. char *comment = (char *)malloc(size + 1);
5. memcpy(comment, src, size);
6. return;
7. }
8. int _tmain(int argc, _TCHAR* argv[]) {
9. getComment(1, "Comment");
10. return 0;
11. }
```

25. 不安全的 API 是导致字符串错误的重要原因, 试列举并说明 3 个不安全的字符串 API?

26. 指出下列代码的错误。

```

1. return y = Ax;
2. int *malloc(int *A, int *x, int n) {
3. int *y = malloc(n * sizeof(int));
4. int i;
5. for (i = 0; i < n; i++)
6. for (j = 0; j < n; j++)
7. y[i] += A[i][j] * x[j];
8. return y;
9. }

```

#### 四、问答题 (24 分, 每题 12 分)

27. 根据输入的不同, 列举并分析下图程序可能的结果?

```

1. bool IsPasswordOkay(void) {
2. char Password[12];
3. gets(Password);
4. if (!strcmp(Password, "goodpass"))
5. return(true);
6. else return(false);
7. }

8. void main(void) {
9. bool PwStatus;
10. puts("Enter password:");
11. PwStatus = IsPasswordOkay();
12. if (PwStatus == false) {
13. puts("Access denied");
14. exit(-1);
15. }
16. else puts("Access granted");
17. }

```

28. 简述 Fuzz 测试的思想, 并描述针对文件的 Fuzz 测试的步骤和流程。