附件 **1：** 试卷格式样张

中国科学技术大学 计算机学院

2019～2020 学年第 二 学期考试试卷

☑ A 卷 □B 卷

课程名称： 信息安全导论 **课程代码：** 011184

开课院系： 计算机科学与技术系 **考试形式：** 闭卷

姓 名： _____ **学 号：** _____ **专 业：** _____

题 号	一	二	三						总 分
得 分									

(以下为试卷正文)

一、填空题(60 个空，每空 1 分，总计 60 分)

1. 习近平指出：没有（ ）就没有国家安全。网络安全指的是网络与信息系统的信息安全。信息安全指信息系统的软件、硬件以及系统中( )的数据受到保护，不因偶然的或者( )原因而遭到破坏、更改、泄露，信息系统连续、可靠、正常地运行，( )不中断。

2. 信息安全的目标是保护网络与信息系统中信息的不可抵赖性和可控性等（ ）属性。( )、( )、( )也称为信息安全的三要素。

3. 所谓信息安全威胁，就是对( )的安全使用可能造成的危害，主要包括意外事件和（ ）两大类。

4. 所谓纵深层防御战略就是采用一个多层次、纵深的( )来保障用户信息及信息系统的安全。在纵深防御战略中，人、技术和(）是三个主要核心要素，要保障信息及信息系统的安全，三者缺一不可。

5. 对称密码体制也叫单钥密码体制或秘密密钥密码体制，非对称密码体制也称为公钥（公开密钥）密码体制。DES 属于（ ）体制，AES 属于（ ）体制，RSA 属于（ ）体制。

6. 在大规模的互联网应用中交换密钥，应该选用（ ）体制；为了验证信息确实来自某个实体，可以采用（ ）技术；为了验证信息的( )，可以在信息的后面附加消息鉴别码。

7. 身份认证是确认某个实体是( )实体的行为，根据被认证实体的不同，

（装订线内不要答题）

身份认证包括两种情况：第一种是计算机认证人的身份，称之为用户认证；第二种是（　　），主要出现在通信过程中的认证握手阶段，称之为认证协议。指纹锁是基于（　　）的用户认证。

8. 所谓信任模型，就是提供用户双方（　　）的框架，是 PKI 系统整个网络结构的基础。通过 X.509 数字证书中的(　　)可以验证 CA 签名证书的合法性，用证书主体的公钥信息加密的信息只能由（　　）解密。

9. 授权是指资源的（　　）准许别的主体以一定的方式访问某种资源，访问控制是(　　)的基础，它控制资源只能按照所授予的权限被访问。Linux 操作系统采用（　　）访问控制策略，多级安全(multilevel secure，MLS)是一种（　　）访问控制策略。具有大量（10000 以上）用户的 Web 应用系统应该选用（　　）的访问控制策略。

10. 空域（　　）和（　　）方法是 2 类典型的信息隐藏技术，离散余弦变换(DCT)是（　　）方法，离散小波变换(DWT) 是（　　）方法。信息隐藏技术的(　　)越高，鲁棒性就越低。
11. Windows 和 Linux 系统达到了 TCSEC 的（　　）安全级别；安全操作系统的安全核是系统中与(　　)的实现有关的部分，包括引用验证机制、(　　)机制、（　　）机制和授权的管理机制等。TCB 在 TCSEC 中的定义是：一个计算机系统中的（　　）的全体。
12. 网络侦察主要包括（　　）和（　　）2 个过程。通过向目标程序的缓冲区写超出其长度的内容，可以造成缓冲区的（　　）。如果目标系统的栈（　　），则缓冲区溢出漏洞（　　）。
13. 为了使局域网内的主机共享一个 IP 地址访问因特网，可以采用（　　）技术；为了保证远程主机到内网的安全访问，可以使用（　　）VPN；为了保证两个局域网穿过因特网进行安全互联，可以使用（　　）VPN。

14. 按照数据来源分类，入侵检测分为 3 类：（　　）（　　）和（　　）。根据检测方法，入侵检测主要分为（　　）和（　　）。

15. 应急响应就是对国内外发生的有关(　　)的事件进行实时响应与分析，提出解决方案和（　　），保证计算机信息系统和网络（　　）。

16. 数字取证的作用，就是通过调查(　　)的计算机和网络系统，收集和保存证据，重建事件，评估事件的状态，(　　)，从而进行犯罪调查或者响应一个计算机安全紧急事件。

17. （　　）是计算机病毒的最基本特征。病毒检测技术主要包括（　　）判定技术和（　　）判定技术。

二、问答题（4 题，每题 5 分，总计 20 分）

1. 简述网络银行保证其根证书可信的一种方法。

2. 简述信息加密与信息隐藏的主要区别。

3. 解释计算机病毒和蠕虫的主要区别。

4.简述信息安全中的机密性、完整性和可用性。

三、程序分析（2 题，10 个空，每个空 2 分，总计 20 分）

1. 32 位 Linux 系统的进程调试及分析

以下列出的是对 32 位 Linux 系统的某个程序进行反汇编及调试的结果。

```
(gdb) disas main
Dump of assembler code for function main:
0x08048328 <main+0>:    push   %ebp
0x08048329 <main+1>:    mov    %esp,%ebp
0x0804832b <main+3>:    sub    $0x28,%esp
0x0804832e <main+6>:    and    $0xfffffff0,%esp
0x08048331 <main+9>:    mov    $0x0,%eax
0x08048336 <main+14>:   sub    %eax,%esp
0x08048338 <main+16>:   sub    $0x8,%esp
0x0804833b <main+19>:   push   $0x8049420
0x08048340 <main+24>:   lea    0xffffffd8(%ebp),%eax
0x08048343 <main+27>:   push   %eax
0x08048344 <main+28>:   call   0x8048268 <strcpy>
0x08048349 <main+33>:   add    $0x10,%esp
0x0804834c <main+36>:   leave
0x0804834d <main+37>:   ret
0x0804834e <main+38>:   nop
0x0804834f <main+39>:   nop
End of assembler dump.
(gdb) b *(main+0)
Breakpoint 1 at 0x8048328
(gdb) b *(main+28)
Breakpoint 2 at 0x8048344
(gdb) b *(main+37)
Breakpoint 3 at 0x804834d
(gdb) r
Starting program: /home/zengfp/ns/ch02/t
  Breakpoint 1, 0x08048328 in main ()
(gdb) display/i $pc
1: x/i $pc   0x8048328 <main>:    push   %ebp
(gdb) x/x $esp
0xbfffde3c:     0x42015574
(gdb) c
Continuing.
Breakpoint 2, 0x08048344 in main ()
1: x/i $pc   0x8048344 <main+28>:         call   0x8048268 <strcpy>
(gdb) x/x $esp
0xbfffde00:     0xbfffde10
(gdb)
0xbfffde04:     0x08049420
(gdb) x/s 0x08049420
0x8049420 <largebuff>:
    "12345123451234512345123451234512345123451234512345===ABCD"
(gdb) c
Continuing.
Breakpoint 3, 0x0804834d in main ()
1: x/i $pc   0x804834d <main+37>:         ret
```

根据以上调试结果填空：

(1) main 的返回地址为(　　)，保存在地址为（　　）的堆栈中。

(2) strcpy 的第一个参数的地址为(　　)；第二个参数的地址保存在地址为（　　）的堆栈中；

(3) 执行 ret 语句后，eip 的值（用字符串表示）为（　　）。

2. 32 位 Windows 进程的调试及分析

在 64 位 Windows10 系统环境用 32 位 C 编译器编译的 C 程序(example.cpp)如下：

```
#include <stdio.h>
#include <string.h>
char largebuff[] ="01234567890123456789012345678901234567890123456789ABCDEFGH";      //48
bytes
int main (void)
{
    char smallbuff[30];
    strcpy (smallbuff, largebuff);
}
```

用 **cl /Fd /Zi /GS- example.cpp** 编译为 32 位可执行程序，用 WinDbg 对可执行程序进行反汇编并跟踪进程的执行，结果如下：

```
0:000> u example!main Lc
example!main [d:\workspace\ns\win32code\src\example.cpp @ 5]:
00826bf0 55              push    ebp
00826bf1 8bec            mov     ebp,esp
00826bf3 83ec20          sub     esp,20h
00826bf6 6800308800      push    offset example!largebuff (00883000)
00826bfb 8d45e0          lea     eax,[ebp-20h]
00826bfe 50              push    eax
00826bff e8aac3ffff      call    example!ILT+8105(_strcpy) (00822fae)
00826c04 83c408          add     esp,8
00826c07 33c0            xor     eax,eax
00826c09 8be5            mov     esp,ebp
00826c0b 5d              pop     ebp
00826c0c c3              ret
0:000> bp example!main
0:000> bp example!main+f
0:000> bp example!main+1c
0:000> g
Breakpoint 0 hit
example!main:
00826bf0 55              push    ebp
0:000> dd esp
    00effd90    00826e3d 00000001 01084490 01086d18
0:000> g
Breakpoint 1 hit
example!main+0xf:
00826bff e8aac3ffff      call    example!ILT+8105(_strcpy) (00822fae)
0:000> dd esp
    00effd64    00effd6c 00883000 00effd90 00effd90

0:000> g
```

Breakpoint 2 hit
example!main+0x1c:
00826c0c c3 ret
0:000> dd esp
00effd90 39383736 44434241 48474645 01086d00

根据以上信息填空：

(1) main 的返回地址为(　　)，保存在地址为（　　）的堆栈中。

(2) strcpy 的第一个参数 smallbuff 的地址为(　　)，第二个参数 largebuff 的地址保存在地址为（　　）的堆栈中；

(3) 执行 ret 语句后，eip 的值（用字符串表示）为（　　）。