《软件安全》

2021-2022 学年下学期期末考试 A 卷

学号：_____姓名：_____成绩：_____

一、选择题（每空 1.5 分，共 30 分）

1、有关软件漏洞，说法错误的是

A、软件漏洞可能在某种情况下被利用来对用户造成恶意攻击

B、黑客产业链的形成与发展不仅危害人民群众的信息、财产等安全，甚至危害国家安全，由此，遏制网络黑色产业的发展、惩治网络犯罪是维护网络安全和社会安全的当务之急。

C、渗透测试就是利用黑客技术对一个目标系统进行渗透攻击的过程，并不考虑目标系统是否会遭受破坏

D、电脑肉鸡是指被控制的远程电脑，往往被植入了木马

2、Oracle 三天前公布了一个新补丁，来弥补一个刚发现的漏洞，该漏洞类型为

A、0day 漏洞　　B、1day 漏洞

C、3day 漏洞　　D、已公开漏洞

3、有关堆，说法错误的是

A、在 Windows 系统中，占有态的堆块被使用它的程序索引，而堆表只索引所有空闲态的堆块。

B、堆表一般位于整个堆区的开始位置，用于索引堆区中所有堆块的重要信息

C、堆是从低地址向高地址扩展的数据结构

D、如果申请 32 字节的堆块，实际会从堆区分配 32 字节的内存

4、当调用函数 g(int a, int b, int c)的时候，变量 a 的地址比变量 c 的地址

A、高　　B、低　　C、相等　　D、不确定

5、汇编指令 MOV AX, 01H; AND AX, 02H 运行之后，AX 寄存器里的值为

A、0　　B、1　　C、2　　D、3

6、发生缓冲区溢出的主要原因是

A、缺少堆栈安全检查机制　　B、栈地址是从低地址向高地址增长

C、未对边界进行检查　　D、栈帧结构设计不合理

7、虚表指针保存在

A、对象的内存空间　　B、静态数据区

C、对象的虚函数中　　D、类的声明中

8、下列不属于对 Shellcode 进行编码的原因的是

A、字符集差异　　B、绕过坏字符

C、绕过安全检测　　D、绕过返回地址

9、下面哪项防护技术是编译器在函数调用和返回时，增加一个 32 位随机数来添加保护和检查功能

A、ASLR　B、GS Stack Protection　C、SEHOP　D、DEP

10、模拟程序运行的过程中，收集程序中的语义信息，探索程序中的可达路径、分析程序中隐藏的错误的方法是

A、模糊测试　B、词法分析　C、符号执行　D、污点分析

11、使用 AFL 对源代码编译插桩，主要的目的是

A、记录代码覆盖率　　B、检测缓冲区溢出

C、获取变量取值　　D、探索路径可达性

12、有关基于数据流的漏洞分析流程，说法错误的是

A、关心的是变量的取值

B、通常需要对程序进行代码建模

C、通过敏感函数的识别和参数分析发现漏洞

D、适合检查因控制流信息非法操作而导致的安全问题

13、下述代码中，关于<12,{z}>的后向切片是

```
1：  int main(){
2：      int x,y,z;
3：      int i=0;
4：      z=0;
5：      y=getchar();
6：      for(;i<100;i++){
7：          if(i%2==1)
8：              x+=y*i;
9：          else
```

10：	z+=1; }
11：	printf(“%d\n”,x);
12：	printf(“%d\n”,z);
13：	}

A、4 10　　B、3 4 6 10　　C、3 4 6 7 9 10　　D、3 4 5 6 7 9 10

14、符号执行中，对于路径约束条件 pc，说法错误的是

A、pc 的初始值为 true

B、pc 表示符号执行中到达一条路径的约束条件

C、根据状态中的 pc 变量确定一次符号执行的完整路径

D、pc 表达式的值不可以求解

15、以下 HTML 页面运行后，页面中将输出

```
<HTML><HEAD></HEAD><BODY>
<?php  $username = “software”;  $SQLStr = “$username=security";  echo
$SQLStr ;  ?>100
</BODY></HTML>
```

A、username=security　　B、usernamesecurity

C、software=security100　　D、software=security

16、如果一个 cookie 设置了 expires 参数，说法错误的是

A、该 cookie 具有一定的生命周期

B、该 cookie 在失效前不会跟随浏览器关闭而销毁

C、该 cookie 只能在创建该 cookie 的电脑里使用

D、该 cookie 失效前会存储在客户端

17、攻击者在用户和 Web 服务器之间截获数据并在两者之间进行转发的攻击方法称为

A、会话保持攻击　　B、会话劫持攻击

C、传输层攻击　　D、中间人攻击

18、关于 XSS，下列说法错误的是

A、启用并使用脚本并不是 XSS 漏洞存在的原因

B、XSS 主要影响的是 Web 应用程序自身

C、反射式 XSS 主要用于将恶意脚本附加到 URL 地址的参数中

D、存储式 XSS 会保存在服务器上，有可能会跨页面存在

19、不能访问文件内容的 PHP 伪协议是

A、php://input　　B、file://　　C、Zip://　　D、Phar://

20、在基于时间的 SQL 盲注中：输入 1’and if(ascii(substr(database(),1,1))>97, sleep(5),

1) 是为了猜解

A、当前数据库名字长度

B、当前数据库名字第一个字符

C、当前数据库中表的个数

D、当前数据库中表的第一个字符

二、判断题（每空 1 分，共 20 分，请用√和×分别表示正确和错误来回答）

1、木马不具传染性，它并不能像病毒那样复制自身。 （ ）

2、对于如下代码：int * p1=new int[100]; char * p2=new char[30]; 则 p1 的值大于 p2 的值。（ ）

3、相对虚拟地址是内存地址相对于文件偏移地址的偏移量。（ ）

4、MOV [BX], 1234H 的寻址方式是相对寻址。（ ）

5、PE 文件中，存放着所使用的动态链接库等外来函数与文件的信息的节是 idata。（ ）

6、CMP EAX, EAX；SETE AL 执行后 AL 寄存器的值为 0。 （ ）

7、消息 HOOK 是一个 Windows 消息的拦截机制，只能用于消息拦截，不能用作 DDL 注入等其他用途。 （ ）

8、DWORD SHOOT 攻击是指能够向任意位置内存写入任意数据的攻击。（ ）

9、触发漏洞、将控制权转移到目标程序的是 shellcode。（ ）

10、考虑程序特定输入的切片方法是动态切片。（ ）

11、污点传播分析隐式流分析的对象是控制依赖关系传播。（ ）

12、下述代码中，函数 func 存在的漏洞类型为整数溢出漏洞。 （ ）

```
void func(char *src){
char buf[256];
int i;
```

```
    for(i = 0;i <= 256;i++)
        buf[i] = src[i];
}
```

13、虚函数的入口地址被统一保存在虚表中。 （ ）

14、单引号法测试注入点的原理是检查 SQL 语法是否正确。 （ ）

15、cookie 在不同浏览器之间不能共享。（ ）

16、文件上传漏洞产生的原因是缺少对一句话木马的检测机制。（ ）

17、存储式 XSS 的安全性会高于反射式 XSS。（ ）

18、Javascript 的作用是在浏览器端增强界面实现，因此基于 Javascript 语言的脚本程序不存在威胁。 （ ）

19、在一个应用中，如果传给 unserialize()的参数是用户可控的，那么攻击者就可以通过传入一个精心构造的序列化字符串，利用 PHP 魔术方法来控制对象内部的变量甚至是函数，因此，PHP 反序列化漏洞又可以称为 PHP 对象注入漏洞。（ ）

20、php://input 可以访问请求的原始数据的只读流，也就是通过 POST 方式发送的内容。借助 PHP 伪协议，攻击者直接将想要在服务器上执行的恶意代码通过 POST 的方式发送给服务器就能完成攻击。（ ）

得分：

三、简答题（共 7 题，34 分）

1、对如下代码

```
#include <iostream>
void why_here(void)
{       printf("why u r here?!\n");
        exit(0);
}
void f()
{         int a = 0
          int value =0;
          int buff;
          int * p = &buff;
          _____= (int)why_here;
}
int main(int argc, char * argv[])
{           f();
            return 0;
}
```

（1）要使得程序运行后显示 why u r here？请完成代码填空。（2 分）

（2）绘制程序运行后显示 why u r here 的栈帧示意图（体现局部变量、返回地址等），并简述显示 why u r here 的原因。（2 分）

2、对于下述代码

```
int formatstring_func2(int argc, char *argv[])
{
      char buffer[100];
      sprintf(buffer, argv[1]);
}
```

（1）编译后运行（Release 模式），并用”aaaabbbbcc%n”作为命令行参数，将会怎么样？（3 分）

（2）产生上述结果的原因是什么？（2 分）

3、简述攻击虚函数的步骤以及可能的攻击策略（5 分）

4、有关符号执行的基本原理，请回答

（1）符号执行的三个关键点（3 分）

（2）符号执行有静态符号执行和动态符号执行，请简述两类分析方法的优缺点。（2 分）

5、简述堆管理结构及 Dword Shoot 攻击的原理（5 分）

6、简述 POST、GET 两种 HTTP 请求方式的作用以及区别。（5 分）

7、对于一个 php 页面，其脚本代码如下：

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == '')
{
        $isempty = true;
} else {
        echo '<pre>';
        echo 'Hello ' . $_GET['name'];
        echo '</pre>';
 }?>
```

（1）指出代码中容易遭受的攻击类型，并简述该种攻击与 SQL 注入攻击的区别。（2 分）

（2）根据上述攻击的特征和利用手法的不同，主要可以分成两大类型。写出这两种类型，并简述两种类型的思想或特点。（3 分）

得分：

四、综合题（共 2 题，16 分）

1. 回答数据流分析有关的问题

```
int contrived(int *p, int *w, int x) {
    int *q;
    if (x) {
        kfree(w);
        q = p;
    }else
        q=w;
    return *q;
}
int contrived_caller(int *w, int x, int *p) {
  kfree(p);
  [...]
  int r = contrived(p, w, x);
  [...]
  return *w;
}
```

（1）简述上述代码中可能出现的漏洞类型（2 分）

（2）要检测上述代码中存在的漏洞，请简述漏洞检测规则（2 分）

（3）请采用路径敏感的数据流分析方法，对上述代码进行建模，并分析哪些路径是安全的，哪些路径是有漏洞的。（4 分）

2. 回答有关符号执行的问题

```
#include <stdio.h>
char u=0;
int main(void)
{
    int i, bits[2]={0,0};
    for (i=0; i<8; i++) {
        bits[(u&(1<<i))!=0]++;
    }
    if (bits[0]==bits[1]) {
        printf("you win!");
    }
    else {
        printf("you lose!");
    }
    return 0;
}
```

对于上述目标程序的 u 的求解，可以使用符号执行求解，代码如下：

```
import angr
import claripy

def main():
p = angr.Project('./issue', load_options={"auto_load_libs": False})
state =
p.factory.entry_state(add_options={angr.options.SYMBOLIC_WRITE_ADDRESSES})

u = claripy.BVS("u", 8)
state.memory.store(0x804a021, u)
sm = p.factory.simulation_manager(state)

def correct(state):
    try:
        return b'win' in state.posix.dumps(1)
        except:
            return False
def wrong(state):
    try:
        return b'lose' in state.posix.dumps(1)
        except:
            return False
```

```
sm.explore(find=correct, avoid=wrong)
return sm.found[0].solver.eval_upto(u, 256)
if __name__ == '__main__':
    print(repr(main()))
```

（1） 简述 claripy.BVS("u", 8)的作用；（2 分）

（2）简述代码 sm.explore(find=correct, avoid=wrong)的作用和工作原理；（2 分）

（3）简述代码 sm.found[0].solver.eval_upto(u, 256)的作用；（2 分）

（4）简述主流约束求解器的两种理论模型。（2 分）