网络空间安全学院 2019 — 2020 学年第 2 学期
《信息系统安全》 考试试卷

A 卷　　开卷　　考试时间：2020 年 6 月 29 日

专业 信息安全 班级 _____ 学号 _____ 姓名

题号	一	二	三	四	五	六	七	总分	核对人
题分	15	30	11	12	10	12	10	100	
得分									

得分	评卷人

一、判断题（判断以下各题的说法是否正确，正确的打 √，错误的的打×，每小题 1.5 分，共 15 分）

（ × ）1. 某项系统功能以共享库的形式提供给用户，由用户按照各自的需要调用，而不是以系统服务的形式提供给全体用户，这种做法主要是考虑最小特权原则。 最小公共机制

（ √ ）2. 自主访问控制 DAC 不能抵御特洛伊木马攻击。

（ √ ）3. BLP 安全模型所指定的原则是利用“不上读”、“不下写”来保证数据的机密性。 3-1 P44

（ × ）4. 系统调用介入（System call interposition）是一种在现代操作系统中加快系统调用处理速度的方法。 隔离 · 4-2

（ × ）5. 用于防止缓冲区溢出攻击的 stack canaries 可以应用于可执行文件，而无需重新编译代码。2-2

（ × ）6. 攻击者不可能在遵循控制流图（CFG）的前提下，进行控制流劫持攻击。 2-3 · P24

（ √ ）7. 使用参数化 SQL 语句实现 Web 应用程序可有效防御 SQL 注入攻击。 6-1

（ × ）8. 最小特权原则可以用来有效的抵御基于缓冲区溢出的攻击。

（ √ ）9. 恶意软件可以通过检测虚拟机管理器 VMM 是否存在，以拒绝在虚拟机环境中运行。4-2

（　　）10.联盟链通常使用 POW 共识机制。

得分	评卷人

二、简答题（简要回答以下问题，每小题 5 分，共 30 分）

1. 在 Linux 操作系统中可以使用 setuid( )系统调用来撤销进程的权限，但在撤销进程权限后，可能存在 capability 泄露问题，请简要说明该问题及其原因。

问题：设 prog 为 root 拥有的 set-uid 程序，由 user 执行
假设程序以 root 打开一个敏感文件 f，
后使用 setuid 撤销权限，降级为 user，
此时实际仍可 write 文件 f.

原因：open 检查权限但 write 不检查，只需有文件描述符即可
根本原因是降级前未清理特权功能，
造成能力泄露（capability leak）

2. 缓冲区溢出攻击和 XSS 攻击都利用不安全的系统设计，即容易受到“Mixing data and code”的影响。对于这两种攻击，请说明用户数据是什么以及它如何影响程序控制的。

缓冲区溢出：数据为输入的文件/字符串等
通过将用户数据溢出到堆栈，程序会将数据作为地址/指令执行，造成 Mixing data and code，后果是控制流劫持与任意代码执行

XSS：用户输入的数据为包含可执行 javascript 代码的输入，XSS 导致服务器返回包含恶意代码的网页，触发浏览器自动执行 js 代码，发送 get/post 请求等，泄露敏感信息，造成任意代码执行

6-1

4-2 1-1，p42

3. 什么是隐蔽通道？它是如何被用于在单个机器上运行的多个相互隔离的 VM 之间泄漏信息的。

隐蔽通道（Covert Channel）是一种在计算机系统中不公开的、未被设计者意图用于通信的通信路径。它可以被用来在不同的进程、用户或系统之间传递信息，而这种通信是不被系统的正常审计和监控机制所检测到的。隐蔽通道的存在可能对系统的安全性构成威胁，因为它们可以被用来绕过安全措施，泄露敏感信息。
在单个机器上运行的多个相互隔离的虚拟机（VM）环境中，隐蔽通道可以被用于以下方式泄漏信息：共享资源利用：虚拟机可能共享某些物理或虚拟资源，如CPU、内存、磁盘或网络接口。攻击者可以通过对这些共享资源的精细操控来传递信息。例如，通过控制对共享内存的访问模式，一台虚拟机可以向另一台虚拟机传递信息。
时间侧信道：通过测量对共享资源的访问时间，一台虚拟机可以推断出另一台虚拟机的活动。例如，如果攻击者能够测量对共享CPU或磁盘的访问延迟，他们可能能够推断出其他虚拟机正在处理的数据。
功耗分析：攻击者可以分析整个系统的功耗模式，来推断出其他虚拟机的活动。不同的计算任务可能具有不同的功耗特征。
网络流量分析：虽然虚拟机之间的网络流量通常是隔离的，但攻击者可能通过分析网络流量模式或时间来传递信息。
声音和电磁辐射：在某些情况下，攻击者可能利用机器发出的声音或电磁辐射作为隐蔽通道。
错误和异常：攻击者可能通过触发特定的错误或异常来传递信息，这些错误或异常在其他虚拟机中可以被检测到。

4. 以下哪种技术可以帮助防御基于堆的控制劫持攻击。简要说明这些的技术如何提供帮助。

2-2

Stackguard, ASLR, DEP, StackShield, CFI

ASLR：随机化堆基址，攻击者很难猜解出可利用内存位置

DEP：标记堆栈内存不可执行，攻击者修改堆内存无效

2-3 CFI：根据 CFG，保证程序遵守 CFG

5. 考虑一个实现电话拨号程序的网站 xyz.com。 当用户输入要拨打的电话号码时，浏览器将打开一个新窗口，其中包含以下定义了监听 postMessage 事件的 Javascript：

```
function receiveMessage(event) {
        // event.data is a phone number from sender
        initiatePhoneCallTo(event.data);
}
window.addEventListener("message", receiveMessage, false);
```

然后，父页面将 postMessage 发送到此窗口，从而激活 receiveMessage 函数以发起呼叫。 请说明恶意网站如何导致访问者向任意电话号码发起电话呼叫（假设访问者已登录到他的 xyz.com 帐户）。

6-1 P32

CSRF.

诱 message 事件监听任意窗口，

放在恶意网站中设定自动执行代码

postMessage，并传入 event.data 为电话号码即可

（可能需要打开题述"新窗口"，获取其

window 对象用于调用 postMessage

6. 以下左图间接跳转会受到一种称为”Branch target injection”的攻击（Spectre 的一种攻击形式），即：jmp 指令查询间接分支预测器（Indirect Branch Predictor），间接分支预测器会依据 BTB（Branch Target Buffer）中内容进行预测，如果预先对 BTB 中的内容进行操纵，那么会使得程序跳转到攻击者给定的目标。以下右图为针对该攻击的防御方法 Retpoline，请简要解释该方法是如何解决 Branch target injection 攻击的？

```
jmp *[eax]
```

```
call l5
l2: pause
lfence
jmp l2
l5: add rsp, 8
push [eax];
ret
```

左边变成右边是retpoline：
在这个例子中，jmp通过rax的值进行间接跳转，如果没有retpoline，处理器会去询问indirect branch predictor,如果之前有攻击者去训练过这个分支，会导致CPU执行特定的一个gadget代码。下面看看retpoline是如何阻止CPU投机执行的。

“1: call load_label ”这句话把 ”2: pause ; lfence ” 的地址压栈，当然也填充了RSB的一项，然后跳到load_label;

“4: mov %rax, (%rsp) ”，这里把间接跳转的地址(*%rax)直接放到了栈顶，注意，这个时候内存中的栈顶地址和RSB里面地址不一样了;

如果这个时候ret CPU投机执行了，会使用第一步填充在RSB的地址进行，也就是 ”2: pause ; lfence ”,这里是一个死循环;

最后，CPU发现了内存栈上的返回地址跟RSB自己投机的地址不一样，所以，投机执行会终止，然后跳到*%rax。retpoline的基本原理，即用一段指令代替之前的简介跳转指令，然后CPU如果投机执行会陷入一个死循环。retpoline就是要绕过这个indirect branch predictor，使得CPU没有办法利用其它人故意训练出来的 分支路径。

Spectre V2漏洞利用CPU的间接分支预测(indirect branch predictor)功能，通过事先训练分支，让分支预测器去影响受害者进程，然后通过侧信道的方式来获取受害者进程的信息。

得分	评卷人

三、（11 分）下表是一个 Linux 目录中的文件信息。

Permissions	Owner	Group	Size	Last Update	File Name
-rws-----x	dave	gdev	1452306	Nov 03 21:11	gtool
drwxrwxrwt	dave	gdev	1452306	Nov 03 21:11	gdata
-rwx--x--x	alice	alice	214768	Nov 03 09:36	setup
-rw-r-----	alice	pcrack	12486	Dec 04 11:00	sourcg
-rw-r--r--	dave	pcrack	14257	Oct 02 18:44	config
-rw--wxr--	root	pcrack	176704	Nov 01 12:23	hosts

一文件夹
任何人可读写执行

（1）列出alice可以write的文件的名称；（2分）

（2）列出dave可以read的文件的名称；（2分）

（3）假设alice执行程序gtool，列出相应进程可以execute的文件的名称；（2分）

（4）假设dave执行程序setup，列出相应进程可以write的文件的名称；（2分）

（5）alice是否可以删除gdata目录内部的文件？为什么？（3分）（提示：sticky位）

(1) setup owner
sourcg owner
(hosts 设alice为pcrack组)

(2) gtool owner
config owner
hosts other
(sourcg 设dave为pcrack组)

原因

(3) gtool 含 setuid，
此时进程 uid = dave
gid不变，设为alice

	原因
gtool	other
setup	other

(4) setup 无 setuid 与 setgid
uid = dave gid = gdave

	原因
gtool	owner
config	owner

(5) 不行. sticky位已设置，
只能新增
只有 uid 与 owner 一致才可以重命名或删除。

4-1 P58

得分	评卷人

四、（12 分）

已知以下一段源代码片段：

```
char tmpbuf[512];
char outbuf[512];
snprintf (tmpbuf, sizeof (tmpbuf), "foo: %s", user);
tmpbuf[sizeof (tmpbuf) - 1] = '\0';
sprintf (outbuf, tmpbuffer);
```

2-2 P35 P10

其中，snprintf 函数定义如下，该函数通过 size 参数来约束格式化字符串的长度：

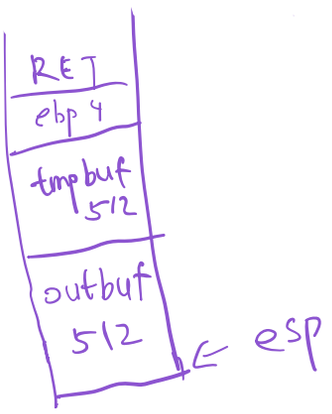
```
int snprintf(char *str, size_t size, const char *format, ...);
```

(1) 分析上述源代码片段存在什么问题，并指出该问题的后果；（4 分）

(2) 请具体说明如何提供字符串 user 来利用该问题，并绕过格式化字符串的长度限制;（4 分）

(3) 在关闭地址随机化且栈不可执行保护未开启的情况下，图示说明构造恶意输入需要完成哪些任务，并说明如何增加跳转到 shellcode 正确地址的机会。（4 分）

(1) 格式化字符串漏洞. user 输入坏串可造成信息泄漏，任意读/写



(2) 占满缓冲区需 512+512+4−len("foo: ") = 1028−5 = 1023

user = %1023d \xaa \xbb \xcc \xdd <nops> <shellcode>

outbuf 溢出，实现常规缓冲区溢出

(3) 收集栈帧地址即 esp 地址

找到合适 ret2libc 拿到 shell

快速 libc 基址，system 偏移等

得分	评卷人

五、（10 分）

Kerberos 涉及三个双向消息交换，一个位于客户端（Client）和密钥分发中心（KDC）之间，一个位于 Client 和票证授予服务（TGS）之间，另一个位于客户端和服务器之间（S） 客户。

（1）简要说明三个消息交换（Client 与 KDC，Client 与 TGS，Client 与 S 之间）的目的；（4 分）

7-1 P80

（2）相对于基于 PKI 的认证系统来说，为什么说 Kerberos 系统的可用性（usability）更好？（2 分）

（3）说明 Kerberos 中是如何实现向客户端认证服务器的？（4 分）

(1) ① 获取一个有效的 $ticket_{TGS}$

② 获取一个有效的 $ticket_V$ ，$K_{c,v}$

③ 认证服务器并获得服务

(2) 证明 重要

(3) Client 与 S 交互，

Client 发送 $ticket_V$ 和 $auth_c$

S 回复 $E_{K_{c,v}}(time_c + 1)$

推理过程：

Client 解出 $time_{c+1}$ ⇒ S 能从 $ticket_V$ 解出 $K_{c,v}$

⇒ S 拥有 K_V

⇒ K_V 由 TGS 签发

⇒ S 为合法服务器

得分	评卷人

六、（12 分）

Seccomp 是在大多数 Linux 发行版中启用的一项内核功能。它对线程所能调用的系统调用进行了限制，限制为：read()，write()，exit()，sigreturn()等 4 个系统调用。如果某线程调用任何其它系统调用，则包括该进程（此线程所在的进程）中的所有线程在内的整个进程都将终止。这种对线程的约束，对于安全性来说是非常理想的。

Chrome 将 Seccomp 用于隔离 HTML 渲染器（HTML Renderers），对 HTML 渲染器的系统调用进行限制，从而使得 HTML 渲染器不会影响 Chrome 控制器。但是，HTML 渲染器需要支持的不仅仅是 Seccomp 允许的四个系统调用。为了解决这个问题，Chrome 在隔离的 HTML 渲染器线程的同一进程地址空间中运行了一个额外的可信线程（Trusted thread）。只要渲染器线程需要进行非授权的系统调用，它就会将请求发送给该可信线程，该线程检查系统调用的参数，如果授权，它则代表渲染器进行调用。通过在同一进程下运行这两个线程，Chrome 在隔离渲染器的前提下实现了良好的整体性能。但是，同一地址空间下运行意味着渲染器线程可以读取和修改可信线程的内存。

假设渲染器线程已经被入侵，并正在运行恶意代码，那么：

（1）假设渲染器线程请求可信线程代表它打开文件，并且文件名存储在内存中。可信线程从内存中读取文件名，验证它是否指向良性文件（例如在/tmp 中），然后发出系统调用以打开文件。 具体来说，进行如下调用：

```
if (benign(filename))
    fp = open(filename, "r");
```

恶意渲染器如何利用此设计打开受保护的文件？ 这种类型的漏洞称为什么？（4 分）

（2）提出一个解决上述问题的方法，并进行解释？（4 分）

（3）上述代码的另一个问题是，渲染器线程和可信线程在相同的地址空间，恶意的渲染器线程可以轻松地操纵可信线程的堆栈。请提出一个解决此问题的方法，并进行解释？（4 分）

(1) TOCTTOU 2-4 P20

(2) 临界区加锁

将操作原子化、消除竞态条件与数据争用

(3) 4-2 隔离-3

软件故障隔离 P7

分	评卷人

（1）如果渲染器线程已经被入侵，它可以在内存中放置一个指向受保护文件的文件名，并请求可信线程代表其打开该文件。由于可信线程会从内存中读取文件名并检查它是否指向一个良性文件，恶意渲染器可以利用这一点，通过覆盖内存中的文件名指针或其指向的值，使其指向一个恶意文件名，从而可能欺骗可信线程打开一个不应该被打开的文件。这种类型的漏洞称为“类型混淆攻击”（Type Confusion Attack）。

（2）为解决上述问题，可以采用以下方法：可信线程在执行系统调用之前，应使用额外的安全检查来验证文件名的来源和上下文，确保文件名没有被篡改，并且确实指向一个良性文件。此外，可以使用安全列表或白名单机制，只允许打开已知安全的文件路径。还可以实施更严格的内存保护机制，如地址空间布局随机化（ASLR）和堆栈保护，以防止恶意代码篡改内存中的文件名。

（3）为解决渲染器线程可能操纵可信线程堆栈的问题，可以采用栈保护技术，如栈溢出保护（Stack Smashing Protector）和堆栈溢出检测。这些技术通过在函数的栈帧中插入保护坎（canary value）或使用其他机制来检测和防止栈溢出。此外，可以使用内存执行保护（Memory Execution Prevention）技术，如数据执行保护（DEP）或地址空间布局随机化（ASLR），以增加攻击者预测和操纵堆栈布局的难度。还可以将可信线程设计为在执行关键操作前进行自我检查，确保其执行环境的完整性未被破坏