1. 信息系统安全概述
   (1) 信息系统安全概况
   (2) 安全攻击示例
   (3) 安全攻击方法概括
   (4) 认识信息系统安全
   (5) 信息系统安全经典要素
   (6) 安全系统基本概念
2. 软件安全
   (1) 控制流劫持及控制流完整性
      ① 控制流劫持攻击
      ② 防御方法简介
      ③ Return to libc 及 ROP
      ④ 控制流完整性（CFI）
   (2) 竞态条件
      ① 竞态条件（Race Condition）漏洞、利用及对策
      ② Dirty COW
      ③ MeltDown & Spectre
3. 信息系统安全模型与机制
   (1) 信息安全模型
      ① 经典安全模型
      ② 自主访问控制
      ③ 强制访问控制
      ④ 基于角色的访问控制
      ⑤ 完整性度量模型
   (2) 安全体系结构
      ① 隔离和最小特权
      ② 访问控制概念
      ③ 操作系统机制
      ④ 浏览器机制
      ⑤ The Confinement Principle 约束原理
      ⑥ 系统调用介入(System Call Interposition)
      ⑦ 基于虚拟机的隔离
      ⑧ 软件故障隔离(Software Fault Isolation)
4. 典型信息系统安全
   (1) 操作系统安全
      ① Unix 系统身份认证
      ② SELinux 访问控制机制
      ③ SELinux 安全机制结构设计
      ④ 80386 保护模式
      ⑤ 系统安全审计
   (2) Web 安全
      ① 命令注入
      ② 跨站请求伪造
      ③ 跨站攻击
   (3) 身份认证
      ① 用户认证
         1) 口令认证，Salt
         2) 挑战-应答(Challenge-response)认证协议
         3) 动态口令
         4) 生物识别技术
         5) 基于令牌(Token-based)的认证
         6) FIDO
      ② 分布式系统中的身份认证
         1) 单点登录系统(Single Sign-On)
         2) 可信中介（KPC 和 CA）

1.调研并阐述 xz-utils 后门的利用原理。
通过供应链攻击的方式劫持 sshd 服务的身份认证逻辑，从而绕过 ssh 身份认证并发送任意代码执行，最终实现远程代码执行攻击（RCE）。
具体来说，首先在 liblzma 库中植入后门代码，因某些 Linux 发行版中 openssh-server 依赖于 libsystemd0，而 libsystemd0 又依赖于 liblzma5。因此，当 sshd 服务启动时，它会间接加载包含后门的 liblzma5.so 库。后门代码中将 sshd 服务中的公钥签名验证函数 RSA_public_decrypt()替换，从而可以绕过 sshd 的身份认证，向目标系统远程发送任意代码执行。

2. 阐述如何利用漏洞程序中对 printf 的错误使用，实现对程序控制流的劫持？
由于 printf 使用格式画字符串来确定如何解释和显示参数，存在格式化字符串漏洞，如果漏洞程序提供输入不受信任的字符串作为参数的机会，则可以精心构造输入泄露内存信息，修改内存值并实现控制流劫持。
具体来说，通过格式化字符串漏洞，可以覆盖栈上的返回地址，将程序重定向到内存中的任意位置，可以是攻击者写入的 shellcode，或者是利用程序原有 gadget 构造的 payload。

3. 阐述如何通过内存泄漏的方式，对 ASLR 防御进行绕过？
可以通过 Got 表泄露函数地址，由于已知版本的 libc 库函数相对地址是固定的，由泄露的函数真实地址可以计算出 libc 基地址，从而可以定位 libc 库函数，并发送 payload 劫持程序控制流，利用 libc 库的提权函数等关键代码获取系统控制，即实施 Ret2Libc 或 ROP 攻击。

4.阐述使用 Ret2libc 方法中多函数调用，分别对 32 位和 64 位漏洞程序进行控制流劫持的原理，及其区别。

Ret2libc 利用程序的栈溢出漏洞来覆盖返回地址，使其指向 libc 库中的代码片段（即有用的 gadgets，可，然后通过多次调用这些函数来执行所需的操作。32 位和 64 位漏洞程序的区别主要是在于函数调用的参数传递方式不同，32 位 linux 完全通过栈来传递参数，但 64 位 linux 函数调用前六个参数是使用 rdi, rsi, rdx, rcs, r8, r9 传递的，由于不是使用栈，所以构建栈上的 ROP 时，需要按顺序构造，把要泄漏的地址 func_got_addr 放入 rdi 寄存器中。

5. 解释为什么普通用户可以利用 Set-UID 程序对特权文件进行修改？
Set-UID（设置用户 ID）是一种 UNIX 和类 UNIX 操作系统中的安全特性，当一个文件被设置了 Set-UID 位时，任何用户执行该程序都会以该文件所有者的权限来运行。所以普通用户使用 set-uid 程序可以获取特权级权限，由此修改特权文件。

6. 分别阐述 Prime+Probe 和 Flush+Reload 两种 Cache 侧信道攻击方法的原理。

Prime+Probe 攻击是一种利用 CPU 缓存一致性机制的侧信道攻击技术。在这种攻击中，攻击者首先（Prime 阶段）使目标数据进入 CPU 的缓存。随后，攻击者探测（Probe 阶段）缓存中的数据是否被访问，这通常通过监测缓存命中或未命中的延迟差异来实现。如果目标数据在缓存中被访问，那么 CPU 处理攻击者指令时会表现出不同的延迟模式，因为缓存一致性机制会确保数据在所有核心之间同步。通过分析这些微小的延迟变化，攻击者可以推断出目标程序的执行情况，从而获取敏感信息。
Flush+Reload 攻击是另一种基于缓存的侧信道攻击技术。在这种攻击中，攻击者首先（Flush 阶段）清空 CPU 缓存中所有数据，确保目标数据不在缓存中。然后，攻击者加载（Reload 阶段）他们自己的数据到缓存中，并监测缓存的行为。如果目标程序随后访问了某些数据，CPU 的缓存一致性机制将导致攻击者加载到缓存中的数据被替换出去，这可以通过监测缓存未命中事件来检测。通过分析缓存未命中的模式，攻击者可以推断出目标程序访问了哪些数据，这可能揭示出密码、密钥或其他敏感信息。

7. 下图是一种 Spectre 的防御方法，请解释其原理。

```
                        call l5
                        l2: pause
jmp *[eax]              lfence
                        jmp l2
                        l5: add rsp, 8
                        push [eax]; put true target on stack
                        ret
```

Spectre 攻击利用了现代处理器中的推测执行（Speculative Execution）特性，通过侧信道攻击获取信息。
上述防御过程调整栈指针后，将 eax 寄存器指向的真实目标地址压入栈中，然后 pause 检查中断，ifence 指令设置同步屏障。代码通过中断和混淆正常的推测执行流程，陷入循环，不去执行预测分支，使得攻击者难以利用预测分支来执行侧信道攻击。