第5章作业

姓名：袁昊男 学号：2018091618008

1、目前已有的ROP检测方法有哪些？存在哪些不足？

**答：**(1)黄志军等人曾提出的DBI的检测方案：取得程序中用到的所有函数的起始以及结束地址，将这些信息存储到HashMap结构中，然后使用工具进行入口地址检测，当所有的入口地址都与HashMap中的数据相对应时，则视为没有ROP攻击，否则当有函数调用时没有从入口地址进入函数时则视为ROP攻击。这种检测方法简单实用，其成立是建立在函数正常跳转时不存在非入口点跳转的现象。但是由于编译器优化等原因导致函数存在正常的非入口点跳转。在其论文中认为正常函数调用时都是从函数的入口地址进入函数的，且在函数入口的前5条指令中不会有ret指令。所以要想构造ROP攻击必然会跳入到函数体的中间位置形成gadget。这种检测思想忽略了函数调用时的非入口点跳转的事实。因此检测的误报率会增高。

(2) 基于API调用特征的检测方法：该机制通过对敏感API函数的调用过程进行合法性检测，从而避免传统的针对大量间接分支的检测，有效地降低运行时开销;通过对攻击者可用的Gadgets进行提取、检测，从而避免CFG构建，降低实现复杂度；通过对API调用特征信息和API调用上下文信息检测，保证程序的安全性。不足之处是：在提取主程序中Gadgets时只考虑了指令片段长度小于等于8的情况，比较聪明的攻击者可能就会选择更长的指令片段实施攻击（指令片段长度越大攻击难度越大，因为长指令片段会产生副作用），对于这种情况本检测方法就会失效。

(3) 细粒度控制流完整性（CFI）检测方案：保护者通过静态分析获取程序的控制流图（CFG），然后为程序的每一个间接分支指令插入检测机制，从而保证程序沿着正确的CFG执行。虽然其能够高效地检测ROP攻击，但是也存在一些问题：首先，程序要对大量的间接分支跳转指令进行验证，这样会导致程序运行时开销增大25%～50%；其次，这种检测机制也可以被绕过，Davi等人提出了一系列的攻击方法并成功绕过CFI。

2、简述DEP机制及对抗方法。

**答：**DEP的主要作用是阻止数据页（默认的堆，栈以及内存池页）执行代码。分为软件DEP和硬件DEP，其中软件DEP就是SafeSEH。而硬件DEP操作系统会通过设置内存页的NX/XD属性标记是否运行在本页执行指令。

对抗方法：

(1) 攻击未启用的DEP程序：并不是只要CPU和操作系统支持DEP，所有程序就安全，对于一些进程来说，只要有某一个模块不支持DEP，就不能开启DEP。在Windows 10下也有许多的程序没有启用DEP。

(2) 利用Re2Libc技术：跳转到ZwSetInformationProcess函数将DEP关闭再转入shellcode执行。原理：一个进程的DEP设置标识保存在KPROCESS结构中的\_KEXECUTE\_OPTIONS 上，这个标识可以通过API函数ZwQueryInformationProcess 和 ZwSetInformationProcess 进行查询和修改。 在系统中找出一处关闭进程DEP的调用，微软有一个LdrpCheckNXCompatibility函数，当出现DLL收到SafeDisc保护的时候（函数中体现为al=1的时候），就会调用ZwSetInformationProcess函数进行关闭DEP，所以可以在调用这个函数前把AL的值改掉，就能够关闭DEP。

(3) 利用Re2Libc技术：跳转到VirtualProtect函数来将shellcode 所在内存页设置为可执行状态，然后转入shellcode执行。原理：在kernel32.dll中，提供了VirtualProtect函数来修改内存属性。

(4) 利用Re2Libc技术：跳转到VirtualAlloc函数开辟一段具有执行权限的内存空间，然后将shellcode复制到这段内存执行。原理：在kernel32.dll中使用VirtualAlloc函数来申请一段具有可执行属性的内存，把第一跳设为Virtual函数地址，然后将shellcode复制到申请的内存空间中。

3、简述检索GS保护机制的实现。

**答：**通过对函数开始和结尾添加代码来阻止典型栈溢出漏洞利用。程序启动时，计算出程序的cookie，将cookie保存在加载模块的.data节中。在函数开始处，cookie将拷贝到栈中，位于返回地址（寄存器EBP）之后、局部变量之前。在函数结尾处程序会把这个cookie和保存在.data模块中的cookie比较，如果不相等，则判定系统栈被破坏。

4、简述ASLR的机制原理和绕过方法。

**答：**从Windows Vista开始引入。通过对堆、栈、共享库映射等线性区域布局的随机化，增加攻击者预测目标地址的难度，防止攻击者直接定位攻击代码位置，阻止漏洞利用。原理如下：

(1) 映像随机化：改变可执行文件和dll文件的加载地址。

(2) 栈随机化：改变线程栈起始地址。

(3) 堆随机化：改变已分配堆的基地址。

(4) 从可执行程序的编译器选项及操作系统加载时地址变化两个方面进行实现和完善：连续选项/DYNAMICICBASE，编译后的程序每次运行，内部栈等结构的地址会被随机化。使用 /DYNAMICICBASE编译的可执行程序和进程中的可执行模块的映像加载基地址实现随机化。

绕过方法：一般系统重启ASLR会生效。许多DLL是由多个系统加载，多次加载时地址不能随机生成，本地攻击可获得所需地址。采用返回地址部分覆盖法：加载库文件的地址空间为8位，可以通过寻找有用的跳转指令，把跳转指令地址的低字节替换栈中的低字节 例：0x12345678，在0x1234XXXX中寻找有用的跳转命令（JMP ESP）。

5、简述SafeSEH的原理。

**答：**编译器在链接生成二进制映像时，把所有合法的异常处理函数的地址解析出来制成一张安全的SEH表，保存在程序的映像的数据块里。当程序调用异常处理函数时会将函数地址与安全SEH表中的地址进行匹配，检测调用的异常处理函数是否位于该表中。若调用的异常处理函数不在SHE表中，则阻止本次调用操作。

6、试从SafeSEH可能存在的弱点，对SafeSEH进行安全性分析。

**答：**SafeSEH是一种有效的漏洞利用防护机制，但存在大量第三方程序和库未使用.Net编译或者未采用/safeSEH连接选项。因此有以下绕过方法：(1) 利用未启用SafeSEH的模块作为跳板；(2) 利用加载模块之外的地址进行绕过.