第三次作业

周玉川

2017221302006

1、 目前已有的 ROP 检测方法有哪些？存在哪些不足？

基于控制流程图(Controlflowgraph，CFG)的 CFI 检测机制	程序按照控制流程在每一个跳转指令进行地址跳转时，通过检查目的地址的有效性，从而保证程序所执行的命令在预先设定的 CFG 范围内。这种检测机制不仅能够检测静态 ROP 攻击还能够检测即时搜索代码的 JIT-ROP 攻击(Just-In-TimeCodeReuse)。
基于地址空间随机化	基于地址空间随机化(AddressSpaceLayoutRandomization，ASLR)能够防御非 JIT-ROP 类型的 ROP 攻击，在 WindowsVista 版本及以后的 Windows 系统中均实现了此技术，但在之前的 Windows 版本并不兼容此方法。

CFI 存在的不足：由于 CFI 针对程序每一步都要进行过滤，所以会产生巨大的性能负担；同样由于该方案是需要对系统的内核程序进行修改，在 Windows8 之前的系统都不能应用这种检测方法。

2、 简述 DEP 机制及对抗方法。

答:

DEP 的提出是为了解决栈溢出的核心问题，防止栈溢出后跳转到 shellcode 执行代码，并区分数据段和代码段，是一种数据执行保护机制。

DEP 的常用对抗方法。

利用 ret2libc 绕过 DEP	不直接跳转到 shellcode，执行库中的代码，被执行的代码可看作是恶意代码
利用 WPM 与 ROP 技术绕过	将 shellcode 写入到不受 DEP 保护的可执行内存中，并成功触发执行
关闭进程的 DEP	系统中存在函数或 API 来启动或关闭 DEP

3、 检索 GS 保护机制的实现。

答:

1) 程序启动时，读取.data 节的第一个 dword
2) 以这个 dword 为基数，通过和当前系统时间，进程 ID，线程 ID，性能计数器进行一系列加密运算（多次 XOR）
3) 把加密后的种子再写入.data 节的第一个 dword
4) 函数在执行前，把加密后的种子取出，与当前 esp 进行异或计算，结果存入“前 EBP”的前面（低地址端）
5) 函数主体正常执行
6) 函数返回前，把 canary 取出与 esp 异或计算后，调用__security_check_cookie 函数进行检查，与.data 节里的种子进行比较，如果校验通过则返回原函数继续执行。如果校验失败，则程序终止。

而且还为额外添加以下操作

1) 加入 cookie。
2) 对栈中变量进行重新排序。
    a) 对函数栈帧进行重新排序，把字符串缓冲区分配在栈帧的最高地址上。
    b) 将函数参数复制到寄存器或放在栈缓冲区上，防止参数被溢出。

4、简述 ASLR 的机制原理和绕过方法。

SALR 原理：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。

绕过方法：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。

5、简述 SafeSEH 的原理。

答:

SafeSEH 原理：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，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，如果 IMAGE 不支持 SafeSEH，则表的地址为 0。

6、试从 SafeSEH 可能存在的弱点，对 SafeSEH 进行安全性分析。

答:

SafeSEH 部分依赖于 DEP,如果绕过了 DEP，SafeSEH 也会被攻击。

因为有大量的第三方库未开启 SafeSEH，所以可以利用未启用 SafeSEH 的模块作为跳板。同样也可以利用加载模块之外的地址进行绕过：如 SHE 中的异常处理函数指向堆区，则通常可以将 shellcode 布置到堆区以触发执行。