程序漏洞&防护

RELRO

在启用延迟绑定时，符号的解析只发生在第一次使用的时候，该过程是通过PLT表进行的，解析完成后，相应的GOT条目会被修改为正确的函数地址。因此，在延迟绑定的情况下，.got.plt必须是可写的，这就给了攻击者篡改地址劫持程序执行的可能。

RELRO（ReLocation Read-Only）机制的提出就是为了解决延迟绑定的安全问题，它最初于2004年由Redhat的工程师Jakub Jelínek实现，它将符号重定向表设置为只读，或者在程序启动时就解析并绑定所有动态符号，从而避免GOT上的地址被篡改。

canary

canary是一种用来防护栈溢出的保护机制。其原理是在一个函数的入口处，先取出一个4字节或者8字节的值存到栈上，当函数结束时会检查这个栈上的值是否和存进去的值一致。

NX (non-execute)

NX的基本原理是将数据所在内存页标识为不可执行，当程序溢出成功转入shellcode时，程序会尝试在数据页面上执行指令，此时CPU就会抛出异常，而不是去执行恶意指令。

PIE

没有引入 PIE 之前，应用程序被执行时只能加载在固定的虚拟地址，我们只能对共享库（shared libraries）或者叫动态库（dynamic libraries）实现所谓的 Position Independent Code。而引入 PIE 后可以允许内核将应用程序本身和那些共享库一样装载在随机的地址，注意这依赖于内核（譬如 Linux）支持所谓的 Address Space Layout Randomization 特性，简称 ASLR。两者结合后，攻击者就很难借助系统中的可执行码实施攻击了，类似缓冲区溢出之类的攻击将无法实施。而且这种提升安全性的代价很小。