网络空间安全导论 · Ch7

计01　容逸朗　2020010869

Q1.

请描述栈溢出攻击和堆溢出攻击的基本原理。

- **栈溢出攻击：**利用缓冲区溢出漏洞来覆盖函数返回地址，从而让程序跳转到攻击者指定的代码。
- **堆溢出攻击：**堆溢出攻击的基本原理是利用堆管理器中的一些漏洞，申请比实际需要更多的内存空间，从而导致堆溢出。攻击者可以通过堆溢出来修改程序的内存布局，改变程序的控制流，执行恶意操作。

Q2.

请简述面向返回地址编程 (ROP) 和全局偏置表劫持攻击 (GOT Hijacking) 的原理，并分析他们能否绕过以下三种内存防御机制，并简述原因：

1. W^X (Write XOR eXecution)
2. ASLR (Address Space Layout Randomization)
3. Stack Canary

- **面向返回地址编程攻击：**攻击原理是利用程序中的一些已知函数来构造恶意代码。具体来说，在 ROP 攻击中，攻击者会利用栈溢出等漏洞，将程序的返回地址修改为另外一个已知函数的地址，从而让程序执行到该函数中。攻击者可以在该函数中构造恶意代码，从而执行攻击操作。
- **全局偏置表劫持攻击：**攻击原理是利用程序中的全局偏置表 (GOT) 来进行攻击。攻击者会利用一些漏洞，将程序中某个函数的 GOT 项指向攻击者构造的代码。这样，当程序执行到该函数时，实际上会跳转到攻击者构造的代码中，从而执行攻击操作。

- 接下来讨论上面的攻击方法能否绕过某些内存防御机制：
- **W^X：** ROP 攻击和 GOT Hijacking 攻击都可以绕过 W^X 机制。
  - 对于 ROP 攻击来说，攻击者的虚假返回地址设置在代码段中，而代码段是存放进程指令的内存区域，必有执行权限。
  - 对于 GOT Hijacking 攻击而言，GOT 表位于可读可写的数据段，同时装载共享库函数的页上必须有可执行权限，因此 W^X 机制不能阻止 GOT Gijacking 政击。
- **ASLR：** ASLR 的功能是随机分配程序的内存地址。因此 ROP 攻击和 GOT Hijacking 攻击都可以绕过 ASLR，因为它们都是利用程序中的已知函数来进行攻击，攻击者不需要知道函数的确切地址。
- **Stack Canary：** ROP 和 GOT Hijackin 都不能绕过 Stack Canary 防护机制。