**软件漏洞分析技术读书报告**

姓名：仝志欣

|  |
| --- |
| **本周预期完成任务：**阅读软件漏洞分析技术的第十一章内容 |
| **本周实际完成进度：**了解基于模式的漏洞分析的方法和过程，及两个典型的工具。 |
| **详细内容：**   1. **基本原理**   **1. 基本概念**  漏洞模式,是研究人员在通过大量已知漏洞的产生原理进行深入分析,并在归纳总结出其一般规律的基础上,抽象出存在安全缺陷的代码段在二进制代码或者汇编代码表现形式上具有的典型特征。**二进制码的漏洞模式分析技术的基本思想来源于源代码的缺陷模式检查技术,两者都是通过模式或者缺陷模型的匹配和检査,实现对漏洞的定位和挖掘。**  基于模式的漏洞分析技术，既可针对软件漏洞进行分析也可针对网络协议漏洞分析，但是网络协议的漏洞往往不具有代表性，本章以软件为主要分析对象。  **2. 二进制文件结构**  要对二进制漏洞模式进行建模和检测,首先要对二进制程序进行逆向分析。在进行逆向分析之前,首先要对研究对象二进制文件有足够的了解。对二进制程序进行逆向分析,旨在剖析程序内部采用的数据结构,以及程序对输入数据的处理流程。由于操作系统的不同，二进制可执行文件的格式有多种，不同的二进制可执行文件格式，结构不同，在进行逆向之前要对几种二进制可执行文件格式有所了解。  一些概念包括 二进制执行格式、UNIX的a.out文件和ELF文件、32位Windows的PE格式。   1. **方法实现** 2. **反汇编分析**   **利用反汇编技术可以将二进制代码转化为可以理解程度更高的汇编级代码；反汇编器是一个静态的工具。**   1. 反汇编的方法   基本算法：第一步，确定区域；第二步，读取该地址（或文件偏移量）所包含的值，并执行一次表查找，将二进制操作码的值与它的汇编语言助记符对应；第三步，获取指令及操作数后，需要对它的汇编语言等价形式进行转化，并将其在反汇编代码中输出；第四步，输出一条指令后继续下一条指令，重复第二、三步，直到反汇编完文件中的所有指令。  当前最主要的算法策略可以分为线性扫描策略和基于控制流的递归扫描策略。   1. 反汇编获取的主要信息   反汇编文本、函数信息、交叉引用关系。   1. 反汇编的不足之处（见于p279） 2. **逆向中间表示** 3. **逆向中间表示的设计**   逆向中间表示的设计原则  寄存器组的设计  寻址方式的设计与翻译  指令系统的设计  而下面的三种语言就是三种经典的面向逆向分析的中间表示：REIL、VEX、Vine   1. 面向逆向分析的中间表示REIL   (3) BitBlaze的汇编级别逆向中间表示VEX    (4) BitBlaze的汇编级别逆向中间表示Vine   1. **漏洞模式建模**   概括来讲，通用性漏洞可抽象出明确的漏洞触发条件，便于采用形式化方式来抽象表示，归于基于形式化表示的漏洞模式；而特定的漏洞可利用描述性的语言结合中间语言来表示，属于基于描述性语言表示的中间漏洞。**漏洞模式的建立是在已知漏洞归纳总结的基础上提出的通用性漏洞触发机制。**  漏洞建模过程中要考虑两个原则：**触发漏洞异常条件的可归纳性、漏洞模式描述方式的适用性**  **下面的缓冲区溢出漏洞和整数溢出类型的漏洞就是通用性漏洞模式，而内存地址对象破坏性调用模式可归类于特定漏洞模式。**   1. 不安全函数调用模式   不安全函数主要包含c函数中的一些没有判断输入长度的内存和字符串操作函数，比如：strcpy、strcat、sprintf等，针对参数量固定和不固定，又分为两类（详细见于p290）   1. 循环写内存模式 2. 整数溢出类漏洞模式   整数溢岀漏洞是由极端输入数据引起的整数溢岀旁路了程序的正常状态检査条件,进而使得接下来的“可信”程序代码段在错误输入数据的控制下,最终发生程序员没有处理也没有预料到)的错误。根据上述定义可知,高危整数溢出漏洞具有如下性质  高危整数溢出漏洞一定出现在一条由输入数据控制的可信指令执行路径上。  高危整数溢出漏洞通常由具有极端数值的输入数据引发。  高危整数溢出漏洞必须引发程序员没有考虑到且没有处理的程序异常。  整数溢出漏洞可能出现在与整型变量相关的运算操作、赋值操作等地方,但不是所有这两类指令都是整数溢出漏洞,还需要判断整型变量的数值是否在其类型所能表示的范围内,以及该数值是否与外部输入相关。与源代码不同的是,二进制程序中没有类型的定义信息,只能从二进制的一些特定操作指令(如SAL,SHL等)得到这些信息。此外,还可以利用相关函数或语句中提取类型信息,类型信息对整型漏洞的检测非常重要。   1. 内存地址对象破坏性调用漏洞模式   内存地址对象破坏性调用的漏洞模式,是指某对象的内存地址保存在一个变量中攻击者通过调用初始化或释放功能的异常操作,将保存对象的内存地址进行释放,从而导致读取变量中保存的内存地址以期访问某对象的过程,会触发内存访问异常。内存地址对象破坏型调用漏洞,属于攻击者了解程序代码后实施的一种非正常调用导致的漏洞,可归类为UAF( use-after-free)漏洞的一种,具体实施过程根据不同的程序代码采用不同的调用方式才能实现,它属于特定漏洞模式,适用于通过下面的描述性语言进行模式的表示。   1. **漏洞模式检测**   漏洞模式可以筛选定位潜在漏洞特征的代码段，这些符合漏洞模式的代码段在可被外部输入控制并且达成特定的约束条件的情况下，才能被触发和利用。为确认漏洞的可触发性，要进行模式抽取，模式抽取的目标对象主要分为**可控变量**和**约束条件**。   1. 缓冲区溢出漏洞模式检测   造成缓冲区溢出漏洞的原因有两个：基于不安全函数的写内存操作和基于循环写内存操作。两种方式的模式检测方法不同。具体可见于图11.7和图11.8 。   1. 整数溢出漏洞模式检测 2. 内存地址对象破坏性调用漏洞模式的检测 3. **典型工具**   **1. BinNavi**  **2. BAP** |
| **相关文件及参考资料：** 软件漏洞分析技术的第十一章内容 |
| **遇到的问题及该阶段学习的想法：虽然本章有许多实例，但仍然感到吃力** |
| **接下来的进度计划：** 阅读第十二章 |