软件安全入门

第九课 软件漏洞挖掘与分析

目录

CONTENTS

01 软件漏洞基础知识

02 软件漏洞机理分析

03 软件漏洞利用

01 软件漏洞基础知识

主要内容包括软件漏洞典型类型、软件漏洞利用基础知识、软件漏洞防护机制基础知识。

漏洞是在硬件、软件、协议的具体实现或系统安全策略上存在的缺陷，导致可被攻击者利用，造成软件研发人员预期之外的安全危害，如敏感信息泄露或被恶意修改，系统崩溃或毁坏，甚至计算机系统和程序被攻击者控制等。

为了准确记录漏洞的各种关键属性信息，管理已经发现的漏洞，指导漏洞的修补工作，需要统一的标准对漏洞进行分类。当前广泛使用的标准如下：

表 9-1　当前被广泛使用的 ITU 漏洞标准

序号	缩写	英文全称	名　称	编　号
1	CVE	Common Vulnerability and Exposure	通用脆弱性与风险标准	ITU-T X.1520
2	CVSS	Common Vulnerability Scoring System	通用脆弱性评分系统	ITU-T X.1521
3	CWE	Common Weakness Enumeration	通用缺陷枚举标准	ITU-T X.1524
4	CWSS	Common Weakness Scoring System	通用缺陷评分系统	ITU-T X.1525
5	OVAL	Open Vulnerability and Assessment Language	开放漏洞评估语言	ITU-T X.1526
6	CPE	Common Platform Enumeration	通用平台枚举标准	ITU-T X.1528

1.2 软件漏洞典型类型

栈溢出漏洞

栈溢出漏洞产生的原因是程序员编写的代码未能检查输入数据是否超出了栈空间的大小，导致向栈中写入了超出其数据区域容纳能力的数据，覆盖了栈中的其他数据，当被覆盖的数据被正常代码引用时，引起程序控制流转移方向，被攻击者劫持。

堆溢出漏洞

堆溢出指的是向堆中写入数据时没有严格检验数据长度，导致写入范围大于堆的大小，覆盖了堆后续的数据。

释放后重用漏洞

已经被释放的对象或内存被某段代码错误地认为还没有释放，从而引起程序错误。攻击者可以在该对象刚刚被释放后通过另一次内存申请获取其访问权限。

整数溢出漏洞

其形成原因在于程序对于整形数据类型的类型转换、比较、计算等操作在编译器实现中的疏漏，利用的主要目标仍然是覆盖栈或堆中的内容。

软件漏洞利用指的是利用软件内部缺陷以实现在该软件的正常运行过程中无法达到的目的或通过精心构造的输入数据达到获取主机控制权、窃取数据、运行可执行代码的目的。

全手工分析

借助于WinDbg、Immunity Debugger等调试工具，通过手工分析生成漏洞利用。
该方法主要依赖分析人员的经验和能力，生成效率低，需投入大量人力和物力。

利用生成工具

使用MetaSploit等漏洞利用生成工具快速生成所需的攻击代码，或使用辅助性工具如Mona、ROPGadget等，搜寻到所有可用于构造ROP的代码片段，通过调试、筛选、调整等步骤，最终形成可利用的攻击代码。

自动生成技术

借助于动态数据流分析方法等各类程序分析方法自动生成漏洞利用。
该方法自动化程度高，但仍有许多问题待解决。

02

软件漏洞机理分析

主要内容包括软件漏洞脆弱点分析、软件漏洞路径分析、软件漏洞内存布局分析等。

栈溢出

栈溢出漏洞的脆弱点是覆盖栈空间的函数或指令如strcpy函数或mov指令。控制流劫持点是引用被覆盖后地址的指令，

堆溢出

堆溢出漏洞按照引起错误的堆相关数据的位置分为两种类型：堆的管理结构和后续堆中的数据。针对堆管理结构的覆盖主要是由于堆块的释放回收算法实现错误引起的。针对后续堆中的数据覆盖主要是堆前一个堆超长的读写造成。

整数溢出

整数溢出的脆弱点通常是两个整数进行计算的指令。控制流劫持点与实际内存布局情况相关。

软件漏洞路径分析的目标是提取程序由数据输入点或程序执行入口到漏洞脆弱点的路径，基于该路径可进一步展开漏洞成因分析、漏洞利用生成等工作。污点传播可以用于追踪数据在漏洞程序中的处理过程：

路径指令序列提取

基于污点传播分析技术提取漏洞路径首先需要确定污点源。污点源应当在输入数据被读入到内存时被引入，即以WSARecv、Recv、ReadFile等输入输出函数的返回位置为污点源引入点。

路径条件提取

漏洞路径条件提取的关键在于确定使控制流转移到脆弱路径的关键节点，以及分析这些节点相互之间的数据依赖和控制依赖关系。

其他情况

(1)指令集支持对于路径分析准确度的影响。
(2)JavaScript等脚本对分析的影响。
污点传播的终止条件：被污点数据所污染的变量是否被用来执行敏感操作。

软件漏洞内存布局分析的目标是：当程序的脆弱点被触发时，确定程序读入数据在内存中的布局情况，分析漏洞利用所需的关键数据所在位置，为漏洞成因分析、漏洞利用生成提供支撑。

输入数入直接映射

指程序通过系统API从外部读入的数据被直接复制到系统内存中，在程序运行到脆弱点时，未发生任何改变。
分析的目标主要针对输入数据读入内存后连续映射区域的地址、长度等基础性信息。

VS

输入数据可逆计算后映射

指输入数据通过系统API读入内存后，当到达程序脆弱点时，已经经过了一定的数学运算。因此除了考虑内存区间的基础信息外，还需要考虑数据的处理过程。对可逆算法实施逆向操作。

03 软件漏洞利用

主要内容包括漏洞攻击链构造、漏洞攻击路径触发等。

漏洞攻击链构造的目标是在内存中找到能够放置控制流转移代码的空间，在放入控制流转移代码后，使得被劫持的控制流能够顺利转移到ShellCode.

漏洞攻击链构造需要确定两方面的内存，一是内存中可利用的区域，二是区域之间的控制流转移关系。

控制流转移关系构造的首要问题是使控制流由劫持点转向攻击链，通常采用两种办法：

（1）**内存喷射**：通过在内存中喷射足够多的重复代码，使得喷射的内存范围足以覆盖被劫持控制流的目标。

（2）**通过已知地址的跳板指令来接管程序执行**：有三种模式的跳板指令：call/jmp register指令、call/jump [register+offset]、连续指令序列。

3.2漏洞攻击路径触发

漏洞攻击路径触发的目标是通过构造输入数据，使程序能够执行到控制流劫持点。

其关键在于：提取程序执行路径中所有与输入相关的条件分支，根据分支判断条件生成相应的约束条件表达式，对表达式进行综合求解，确定输入数据的取值范围。

可以利用在污点传播图中寻找污点分支判断指令节点，以此节点为起点，在污点传播图中进行回溯。

课程小结

- **1.软件漏洞基础知识**
- 主要内容包括：软件漏洞典型类型、软件漏洞利用基础知识、软件漏洞防护机制基础知识。
- **2.软件漏洞机理分析**
- 主要内容包括：软件漏洞脆弱点分析、软件漏洞路径分析、软件漏洞内存布局分析等。
- **3.软件漏洞利用**
- 主要内容包括：漏洞攻击链构造、漏洞攻击路径触发等。

谢谢观赏