**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六章内容 |
| **本周实际完成进度：**认识到符号执行技术的几个作用；了解符号执行技术相关概念；能够结合实例分析符号执行技术的实现；了解两款典型的应用符号执行技术的漏洞分析工具。 |
| **详细内容：**   1. **基本原理**   **1. 基本概念**  符号分析过程：将程序中一些需要关注但又不能直接确定取值的变量用符号表示其取值，然后分析可能执行的流程，将程序中变量的值表示为符号和变量的计算表达式。判断路径的可满足性；约束求解器；构造程序的测试用例；与其他技术相结合时进行排除不可能的程序路径从而降低漏洞分析的误报率。  从测试生成角度：符号执行可以生成高覆盖面的测试用例；从漏洞分析角度：可以生成触发漏洞的具体输入，漏洞分析人员可以利用符号执行工具生成的输入验证并分析该漏洞。  **2. 检测程序漏洞**  检测过程的工作原理与数据流分析相似，只不过在漏洞分析规则里换为符号标记规则和取值约束，在静态分析里换成了符号执行和约束求解的交替执行。  ①正向的符号执行：通过正向遍历程序路径的方式进行程序漏洞的分析。通过分析路径上的程序语句，不断将变量的取值表示为符号和常量的表达式，将路径条件表示为符号的约束，同时对符号在路径上需要满足的取值约束进行求解，判断路径是否可行。  ②逆向的符号执行：更加具有针对性。直接在关键点分析所关心的变量是否满足存在漏洞的约束条件，通过逆向分析不断获取路径条件对所关心的变量的取值约束，并计算关心的变量在当前的约束下是否还能满足漏洞的约束条件。  3．构造测试用例  每一步计算表达式过后，将路径条件转化为对符号取值的约束并记录下来。排除不可能的路径。最后根据输入限制，构造输入测试用例  4. 与其他技术相结合  一方面，路径可行性分析可以排除不可能执行的路径，减小分析范围，是结果更精确；另一方面，路径可行性分析减少漏洞分析结果中可能会有一些不可能被执行路径的相关漏洞，减少误报。无论是漏洞分析过程中还是验证结果中使用符号执行和约束求解，最终目的都是通过判断路径的可行性，减少漏洞的误报，辅助生成漏洞的触发条件。  路径可行性分析  疑似漏洞的验证环节   1. **方法实现—基本原理的实现** 2. **使用符号执行检测程序漏洞**   将漏洞存在的条件表示为符号取值的约束。根据约束进行求解。  (1)正向的符号执行  a．过程内分析  具体的程序语句决定了要怎样进行分析。  ①声明语句分析  记录分配变量的储存空间大小；另一个目的是发现程序中的全局变量  ②分析赋值语句  x=x+1;a[i]=j;而对于指针变量有关的赋值语句，常常用于指向数组中，通过分析指针地址是否超过数组范围来确定是否有漏洞。  ③分析控制转移语句  控制转移路径其实是增加了符号变量约束，同时可能增加了路径，但要对增加的路径进行约束求解，用到约束求解器。但是也应看到约束求解器可能并不能求解很复杂的语句，当路径条件所对应的符号约束不能求解时，常常选择继续分析该路径。  分析调用语句  其一，过程调用将会引入新的符号：调用的变量、如果对指针分析要有指针变量的取值、函数、命令行参数、函数参数；其二，变量被分配的内存要有记录；对一些关键函数要对使用情况进行检查，比如粘贴字符串  b．过程间分析  两种遍历方式：深度遍历、广度遍历。  深度遍历：不使用摘要（每一次分析相同摘要时可能在不同的路径下）  广度遍历：在队列中记录一些需要分析的变量的符号表示和符号取值的路径约束条件，并且根据队列调度对基本快进行分析。  循环结构：直观上直接根据代码所示的循环次数对代码分析相应的次数；如果次数很多考虑分析少数次就退出；无法得知循环次数时考虑只执行一次。  (2)逆向的符号分析  先找一个怀疑点，然后逆向分析。过程：得到漏洞存在的取值约束，往上走如果存在路径约束或者取值约束就加上，遇到赋值语句可以代换，直到遇到以下两种情况。一种是分析过程中发现存在漏洞的约束不满足，即无漏洞，一种是分析到达程序变量的入口点，说明存在漏洞。  分析过程中由于常常无法同时进行别名分析和指向分析，所以可以选择在进行逆向分析前先进行别名分析或者指向分析。  约束不可解选择继续分析。   1. **使用符号执行构造测试用例**   就是找出满足路径可行的一系列可行值，这些不能确定的变量组成的表达式，和路径条件一起组成约束条件。当构造出来的取值范围满足这些约束条件，则测试用例就出来了。   1. **典型工具**   **1. Clang**  **2. KLEE** |
| **相关文件及参考资料：** 软件漏洞分析技术的第六章内容 |
| **遇到的问题及该阶段学习的想法：** |
| **接下来的进度计划：** 阅读第七、八两章。 |