软件脆弱性分析与漏洞利用

秦子茗[1]、杨锃涛[2]、刘迅[3]

摘要：软件的漏洞一直是让软件工程师们头疼的难题。如何在编辑软件的过程中减少或者消除漏洞的出现，一直是人们所关心的事。本文拟从多个方面尝试论证软件漏洞的产生不可避免，只能通过一定的手段做到尽量少的漏洞出现。

引言：随着信息技术的发展，越来越多的软件开始出现在人们的视野中。它们有的是善意的,有的是恶意的。有的为人们提供了社交平台,有的方便了人们生活，而有的却是为了获得不法信息。而无一例外，这些软件或多或少都存在漏洞。这些漏洞有的无关紧要，并不会让软件出什么问题，但是有的却可能造成几百甚至上千万的经济损失。

1 软件漏洞成因

1.1 软件开发人员存在的缺陷

一个软件的产出所需要的代码量是非常庞大的，而往往有些功能模块是不同软件所共有的。软件编写人员为了减少工作量，直接进行了简单的复制粘贴，使得原来可能只在一个软件的漏洞被直接嫁接到了许多软件内。

单论代码本身，受限于软件编写人员的水平与安全意识，往往会出现包括但不限于数组越界、堆缓冲区溢出、内存泄露等一系列问题。而这些问题有的是可以通过调试实现解决，而有些或许会出现复杂的情况。例如，内存错误，编译器不能够自动发现这些错误，而且错误的发生具有偶然性，也许调试好的软件，到了客户的机器上就出现问题了。这些问题出现的主要原因有内存分配未成功却使用了它；内存分配虽然成功，但是尚未初始化就引用它；内存分配成功并且已经初始化，但是操作越过了内存的边界。内存分配成功并且已经初始化，操作也未超越内存的边界，但是没有释放内存资源，造成了内存泄露。

软件编写人员往往会认为用户输入的数据是符合规则的，所以在编写代码时会缺少对于输入参数的检测，这又是一个可能被攻击者利用的漏洞。

1.2 软件自身存在的缺陷

绝大部分软件的开发，最终目的是为了商业或是其他服务业的功能实现。所以在软件的设计之初,软件设计人员经常注重的是软件的功能而忽视了安全性的因素。实际上，安全性和功能之间是对立的，所以这就要求软件的开发人员进行合理的取舍。许多的软件安全措施通常对于使用者会产生一定的复杂性，对于软件的功能强大的性质可能会被乱用。

而随着软件的不断更新和用户反馈给软件产品开发人员的使用信息使得软件自身的特征不断地增加，在这些特征之间相互的交互方式越来越复杂。某些特

---

[1] 负责第二部分
[2] 负责第一部分、摘要、引言
[3] 负责第三部分

征交互对系统有益且是必要的，软件设计和开发人员希望通过这些交互来达到某种目的，满足用户的需求。而另外一些特征交互是非预期的，它们往往源于分析、设计或实现中的某种缺陷。非预期的特征交互可能破坏系统的稳定性或状态一致性，严重的甚至会导致系统崩溃。

软件的向后兼容性和软件升级后的安全缺陷通常是对立的。例如软件的前一个版本中如果有漏洞被发现，那么在这个软件之后的版本就会更改修复这个漏洞，这就是向后兼容性的表现方式。所以软件版本的更新在某种程度上会对向后兼容性做出一定的考虑，这就有可能使用的工作方式是不安全的。这就有可能出现一些漏洞给那些攻击者提供机会。

结语：

软件漏洞的出现是不可避免的，我们能做的只是在漏洞发现后尽快将其填补，尽可能的降低不法分子利用漏洞的时间，并且加强软件开发人员的安全意识，对不同类型的软件采取不同级别的安全排查。

2 软件漏洞如何被发现

软件漏洞不仅会影响软件的正常运行，而且还会被攻击者利用来获取目标软件所在系统的授权或者非授权执行某些程序从而发起针对软件的攻击。这些攻击往往以网络犯罪或窃取机密为目的，对社会和国家安全造成重大威胁。而上述攻击行为的关键前提便是应用或系统软件漏洞的发现，即“漏洞挖掘”。

软件漏洞一部分是开发时由于程序员经验不足或者疏忽而留下的安全隐患，另一部分是软件在逻辑设计上的缺陷。近些年来，漏洞数量呈上升趋势，新漏洞从公布到被利用的时间也越来越短。除了利用已知漏洞，黑客们也善于挖掘并利用一些尚未公布的漏洞，发起病毒攻击，或出售漏洞资料，满足经济目的。于是，漏洞挖掘也成为网络安全人员主动发现并修补软件缺陷，提升内构安全的重要方式。下面笔者将介绍几种常见的漏洞挖掘技术并分析其优劣。

2.1 静态漏洞挖掘技术

静态分析技术是对被分析目标的源程序进行分析检测，发现程序中存在的安全漏洞或隐患，是一种白盒分析技术。主要包括源代码扫描、静态污点分析、可达路径分析、静态符号执行等技术。它的方法主要包括静态字符串搜索、上下文搜索。静态分析过程主要是找到不正确的函数调用及返回状态，特别是可能未进行边界检查或边界检查不正确的函数调用，可能造成缓冲区溢出的函数、外部调用函数、共享内存函数以及函数指针等。

对开放源代码的程序，通过检测程序中不符合安全规则的文件结构、命名规则、函数、堆栈指针可以发现程序中存在的安全缺陷。被分析目标没有附带源程序时，就需要对程序进行逆向工程，获取类似于源代码的逆向工程代码，然后再进行搜索。使用与源代码相似的方法，也可以发现程序中的漏洞，这类静态分析方法叫做反汇编扫描。由于采用了底层的汇编语言进行漏洞分析，在理论上可以发现所有计算机可运行的漏洞，对于不公开源代码的程序来说往往是最有效的发现安全漏洞的办法。

静态分析技术能够高效快速地完成对程序代码的检查，并且其代码覆盖率较高、漏报较少。然而，由于静态分析缺乏运行时的数据，并且缺乏动态的测试过程和针对细节的安全评估，导致静态分析技术准确率较低、误报较多。

2.2 动态漏洞挖掘技术

动态分析技术起源于软件调试技术，是用调试器作为动态分析工具，但不同于软件调试技术的是它往往处理的是没有源代码的被分析程序，或是被逆向工程过的被分析程序。主要包括 Fuzzing 测试，动态污点分析，动态符号执行等技术。

目前工业界使用最为广泛的自动化漏洞挖掘技术是模糊测试，也称“fuzzing 技术”。Fuzzing 技术是一种基于缺陷注入的自动软件测试技术，是一种黑盒分析技术方法，使用大量半有效数据作为应用程序的输入，以程序是否出现异常为标志，来发现应用程序中可能存在的安全漏洞。半有效数据是指被测目标程序的必要标识部分和大部分数据是有效的，有意构造的数据部分是无效的，应用程序在处理该数据时就可能发生错误，导致应用程序的崩溃或者触发相应的安全漏洞[3]。

与其它漏洞挖掘技术相比，Fuzzing 技术具有思想简单，容易理解、从发现漏洞到漏洞重现容易、不存在误报的优点。同时它也存在黑盒分析的典型缺点：测试结果的准确性取决于测试用例的设计，而且具有不通用、构造测试周期长等问题。除此之外，传统模糊测试存在对目标程序缺乏了解，对程序自身提供的信息利用不充分，测试用例的生成具有一定的盲目性，代码覆盖率低等缺点，有待进一步改善。

常用的 Fuzzer 软件包括 SPIKE Proxy、Peach Fuzzer Framework、Acunetix Web Vulnerability Scanner 的 HTTP Fuzzer、AFL 等。

动态污点分析技术对程序的污点数据在系统程序中的传播进行实时监控，在关键的程序点检查关键的操作是否会受到污点信息的影响。动态符号执行技术将无法确定取值的关键变量用符号表示取值，逐步分析程序执行流程，程序中变量的取值表示为符号与参数的表达式。

动态分析技术具有较高的漏洞挖掘准确率，但动态分析技术代码覆盖率相对较低，条件不满足时代码将无法执行，同时存在漏报问题。考虑到静态测试可以覆盖所有代码，所以目前结合静态和动态分析进行漏洞挖掘成为主流的安全漏洞研究方式。

常见的动态分析工具有 SoftIce、OllyDbg、WinDbg 等。

2.3 智能化赋能漏洞挖掘技术

随着人工智能相关研究的蓬勃发展，机器学习的应用越来越广泛，其在漏洞挖掘领域同样大受关注。目前的研究主要聚焦于利用机器学习筛选包含漏洞的程序，近年来研究人员采用各种学习算法构建了多种漏洞挖掘模型。

静态污点分析技术空间使用率高，误报率高，而机器学习能够快速处理大量的样本，结合机器学习来降低误报率则是一种可行的方法。符号执行的一个关键问题是路径执行空间爆炸问题，通过机器学习确定可疑函数集合，利用可疑函数集合来指导符号执行能够有效减少路径数量，减缓路径执行空间爆炸问题，提高了符号执行的性能[4]。Fuzzing 测试需生成更有效的测试样例才能有效地触发漏洞，结合机器学习能够提高 Fuzzing 测试的效果。同时利用人工智能技术能够有效地提升代码覆盖率，进而有效地进行漏洞挖掘。由此可见，机器学习不仅能够用于挖掘漏洞，而且其分类结果也可以用来指导程序分析技术进行漏洞挖掘，提升漏洞挖掘效率。将机器学习同程序分析技术结合来解决静态分析高误报、低准确率、高动态分析漏报率、低代码覆盖率等问题，这为缓解空间利用率高、约束求解难、路径执行空间爆炸等问题提供了新的思路，将机器学习同静态分析技

术、动态分析技术结合进行漏洞挖掘也是一个可探讨的方向[5]。

3 利用漏洞实施网络攻击

对于攻击者来说，仅仅发现、掌握软件中的漏洞这并不够，只有具体到在目标攻击对象的软件上利用这些漏洞实施攻击，对于攻击方来说才是有价值的。在互联网时代，网络便成为了实施这种行为的场所。相较于传统空间，网络空间中的攻击更为隐蔽和复杂。这个场景下，攻击者利用漏洞，通过网络实施针对目标的攻击。对于这一问题的讨论，可以从软件的生命周期角度分为以下几个环节。

3.1 开发工具

软件开发者在开发软件之始，就有可能出现问题。开发者有可能使用了有问题的开发工具，导致开发出来的软件存在后门或者是其他恶意代码。XcodeGhost 就是最直观的例子。开发工具 Xcode 被攻击者修改后在非官方渠道发布，经由这个 Xcode 编译出来的苹果程序会被植入后门。通过设置接收服务器，攻击者可以对被感染软件的用户远程获取个人数据、执行恶意代码。

3.2 源代码攻击

在开发的过程中，源代码本身作为最核心的内容，也有可能受到攻击。攻击者可以利用窃听系统窃取源代码内容，也可以通过 LPE 实现本地权限提升，篡改甚至删除源代码。

3.3 下载网站攻击

现今，软件的分发更多依靠网络，而不是过去传统的硬盘软盘拷贝等模式。那么其所依赖的提供下载服务的下载服务器就成了问题的关键。攻击者可以利用 DNS 污染等技术将用户指向一个伪造的下载网页，其中包含的是被嵌入了恶意代码的软件文件。同时我们应当注意到，网页背后的服务器也是由软件构成的。攻击者还可以通过对服务器软件的攻击、对下载提供商的 DDoS 攻击阻止用户获取目标软件，进而达到网络攻击的目标。

结语：本文探讨了软件漏洞的可能成因、软件漏洞的挖掘技术以及漏洞利用的几种可能性。现实中的漏洞成千上万，本文无疑做不到对所有软件漏洞做出一个全面的概括和掌握，但我们应当注意到，掌握万千现象背后的规律以及得到规律的思维方式是更重要的。具体的漏洞将是作为例子，有助于我们把握理解表象后的更一般规律。

参考文献：

[1] 刘峪杰. 试谈软件漏洞的分类[J]. 电脑编程技巧与维护,2011(12):112-113,123.

[2] 吕金和. 由指针和数组带来的软件安全性缺陷[J]. 计算机安全,2010(5):64-67.

[3] 黄影，邹颀伟，范科峰 基于 Fuzzing 测试的工控网络协议漏洞挖掘技术[J]. 通信学报,2018,(39),254-261

[4] 邹权臣，张涛，吴润浦，马金鑫，李美聪，陈晨，侯长玉. 从自动化到智能化: 软件漏洞挖掘技术进展[J]. 清华大学学报,2018,58(12)
[5] 孙鸿宇，何远，王基策，董颖，朱立鹏，王鹤，张玉清. 人工智能技术在安全漏洞领域的应用[J]. 通信学报,2018,39(8),1-17