

# 《模糊测试》

## 图书基本信息

书名：《模糊测试》

13位ISBN编号：9787111257554

10位ISBN编号：7111257553

出版时间：2009-1

出版社：斯顿 (Michael Sutton)、Adam Greene、Pedram Amini、黄陇 机械工业出版社 (2009-01出版)

作者：[美] Michael Sutton,Adam Greene,Pedram Amini

页数：363

译者：黄陇,于莉莉,李虎

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：[www.tushu111.com](http://www.tushu111.com)

# 《模糊测试》

## 前言

安全漏洞是研究安全问题的生命线。无论是执行渗透测试、评价新产品还是审核关键构件的源代码，安全漏洞都驱动着我们的决策，让我们有理由花费时间，并且很多年来一直影响着我们的选择。源代码审核是一种白盒测试技术，这是一种很长时间以来都流行的软件产品安全漏洞检测方法。这种方法需要审核者了解编程概念和产品功能的每一个细节，深入洞察产品的运行环境。除此之外，源代码审核还有一个显而易见的缺陷——必须首先要获得产品的源代码。幸运的是，除了白盒技术外，我们还可以使用不需要访问源代码的黑盒技术。模糊测试就是黑盒技术中的一种可选方法，这种方法在发掘那些用审核方法无法发现的产品关键安全漏洞方面被证明是成功的。模糊测试是这样的一个过程：向产品有意识地输入无效数据以期望触发错误条件或引起产品的故障。这些错误条件可以指导我们找出那些可挖掘的安全漏洞。

# 《模糊测试》

## 内容概要

《模糊测试强制性安全漏洞发掘》主要内容：模糊测试的工作原理，模糊测试相比其他安全性测试方法的关键优势，模糊测试在查找网络协议，文件格式及Web应用安全漏洞中的技术现状等。演示了自动模糊工具的用法，并给出多个说明模糊测试强大效力的历史案例。

# 《模糊测试》

## 作者简介

Michael Sutton是SPI Dynamics公司的安全布道师。他还是Web应用安全组织(WASC)的成员，负责其中的Web应用安全统计项目。

Adam Greene目前担任纽约某大型金融新闻公司的工程师。此前他曾经是iDefense公司的工程师，这是位于Reston, VA.的一家智能技术公司。Adam Greene在计算机安全领域的主要研究兴趣是可靠挖掘方法、模糊测试和基于UNIX系统的审核和挖掘开发。

Pedram Amini是Tipping Point公司的安全研究和产品安全评估组的项目领导。此前他曾经是iDefence实验室的主任助手，同时也是该实验室的创建者之一。他的主要兴趣是研究逆向工程——开发自动支持工具、插件和脚本。

这三位作者经常出席Black Hat安全大会并在其中做主题报告。

## 书籍目录

- 译者序
- 译者简介
- 序言
- 前言
- 致谢
- 第一部分 背景
- 第1章 安全漏洞发掘方法学
  - 1.1 白盒测试
    - 1.1.1 源代码评审
    - 1.1.2 工具和自动化
    - 1.1.3 优点合缺点
  - 1.2 黑盒测试
    - 1.2.1 人工测试
    - 1.2.2 自动测试或模糊测试
    - 1.2.3 优点和缺点
  - 1.3 灰盒测试
    - 1.3.1 二进制审核
    - 1.3.2 自动化的二进制审核
    - 1.3.3 优点和缺点
  - 1.4 小结
- 第2章 什么是模糊测试
  - 2.1 模糊测试的定义
  - 2.2 模糊测试的历史
  - 2.3 模糊测试阶段
  - 2.4 模糊测试的局限性和期望
    - 2.4.1 访问控制缺陷
    - 2.4.2 设计逻辑不良
    - 2.4.3 后门
    - 2.4.4 内存破坏
    - 2.4.5 多阶段安全漏洞
  - 2.5 小结
- 第3章 模糊测试方法和模糊器类型
  - 3.1 模糊测试方法
    - 3.1.1 预先生成测试用例
    - 3.1.2 随机方法
    - 3.1.3 协议变异人工测试
    - 3.1.4 变异或强制性测试
    - 3.1.5 自动协议生成测试
  - 3.2 模糊器类型
    - 3.2.1 本地模糊器
    - 3.2.2 远程模糊器
    - 3.2.3 内存模糊器
    - 3.2.4 模糊器框架
  - 3.3 小结
- 第4章 数据表示和分析
  - 4.1 什么是协议
  - 4.2 协议域

- 4.3简单文本协议
- 4.4二进制协议
- 4.5网络协议
- 4.6文件格式
- 4.7常见的协议元素
  - 4.7.1名字-值对
  - 4.7.2块标识符
  - 4.7.3块长度
  - 4.7.4校验和
- 4.8小结
- 第5章 有效模糊测试的需求
  - 5.1可重现性和文档记录
  - 5.2可重用性
  - 5.3过程状态和过程深度
  - 5.4跟踪、代码覆盖和度量
  - 5.5错误检测
  - 5.6资源约束
  - 5.7小结
- 第二部分 目标和自动化
- 第6章 自动化测试合测试数据生成
  - 6.1自动化测试的价值
  - 6.2有用的工具和库
    - 6.2.1ETHERREAL / WIRESHARK
    - 6.2.2LIBDASM和LIBDISASM
    - 6.2.3LIBNET / LIBNETNT
    - 6.2.4LIBPCAP
    - 6.2.5METRO PACKET LIBRARY
    - 6.2.6PTRACE
    - 6.2.7PYTHON EXTENSIONS
  - 6.3编程语言的选择
  - 6.4测试数据生成和模糊启发式
    - 6.4.1整型值
    - 6.4.2字符串重复
    - 6.4.3字段分隔符
    - 6.4.4格式化字符串
    - 6.4.5字符翻译
    - 6.4.6目录遍历
    - 6.4.7命令注入
  - 6.5小结
- 第7章 环境变量和参数的模糊测试
  - 7.1本地化模糊测试介绍
    - 7.1.1命令行参数
    - 7.1.2环境变量
  - 7.2本地化模糊测试准则
  - 7.3寻找目标程序
  - 7.4本地化模糊测试方法
  - 7.5枚举环境变量
  - 7.6自动化的环境变量测试
  - 7.7检测问题

## 7.8小结

## 第8章 环境变量和参数的模糊测试自动化

### 8.1iFUZZ本地化模糊器的特性

### 8.2iFUZZ的开发

### 8.3iFUZZ的开发语言

### 8.4实例研究

### 8.5益处和改进的余地

### 8.6小结

## 第9章 Web应用程序和服务器的模糊测试

### 9.1什么是Web应用程序模糊测试

### 9.2目标应用

### 9.3测试方法

#### 9.3.1建立目标环境

#### 9.3.2输入

### 9.4漏洞

### 9.5异常检测

### 9.6小结

## 第10章 Web应用程序和服务器的模糊测试：自动化

### 10.1 Web应用模糊器

### 10.2WebFuzz的特性

#### 10.2.1请求

#### 10.2.2模糊变量

#### 10.2.3响应

### 10.3必要的背景知识

#### 10.3.1识别请求

#### 10.3.2漏洞检测

### 10.4 WebFuzz的开发

#### 10.4.1开发方法

#### 10.4.2开发语言的选择

#### 10.4.3设计

### 10.5实例研究

#### 10.5.1目录遍历

#### 10.5.2溢出

#### 10.5.3SQL注入

#### 10.5.4XSS脚本

### 10.6益处和改进的余地

### 10.7小结

## 第11章 文件格式模糊测试

### 11.1目标应用

### 11.2方法

#### 11.2.1强制性或基于变异的模糊测试

#### 11.2.2智能强制性或基于生成的模糊测试

### 11.3输入

### 11.4漏洞

#### 11.4.1拒绝服务

#### 11.4.2整数处理问题

#### 11.4.3简单的栈和堆溢出

#### 11.4.4逻辑错误

#### 11.4.5格式化字符串

11.4.6竞争条件

11.5漏洞检测

11.6小结

第12章 文件格式模糊测试：UNIX平台上的自动化测试

12.1NOTSPIKEFILE和SPIKEFILE

12.2开发方法

12.2.1异常检测引擎

12.2.2异常报告(异常检测)

12.2.3核心模糊测试引擎

12.3有意义的代码片段

12.3.1通常感兴趣的UNIX信号

12.3.2不太感兴趣的UNIX信号

12.4僵死进程

12.5使用的注意事项

12.5.1ADOBE ACROBAT

12.5.2REALNETWORKS REALPLAYRE

12.6实例研究：REALPLAYER REALPIX格式化字符串漏洞

12.7语言

12.8小结

第13章 文件格式模糊测试：Windows平台上的自动化测试

13.1 Windows文件格式漏洞

13.2 FileFuzz的特性

13.2.1创建文件

13.2.2应用程序执行

13.2.3异常检测

13.2.4保存的审核

13.3必要的背景知识

13.4 FileFuzz的开发

13.4.1开发方法

13.4.2开发语言的选择

13.4.3设计

13.5实例研究

13.6益处和改进的余地

13.7小结

第14章 网络协议模糊测试

14.1什么是网络协议模糊测试

14.2目标应用

14.2.1数据链路层

14.2.2网络层

14.2.3传输层

14.2.4会话层

14.2.5表示层

14.2.6应用层

14.3测试方法

14.3.1强制性或基于变异的模糊测试

14.3.2智能强制性模糊测试和基于生成的模糊测试

14.3.3修改的客户端变异模糊测试

14.4错误检测

14.4.1人工方法(基于调试器)



14.4.2 自动化方法(基于代理)

14.4.3 其他方法

14.5 小结

第15章 网络协议模糊测试：UNIX平台上的自动化测试

15.1 使用SPIKE进行模糊测试

15.1.1 选择测试目标

15.1.2 协议逆向工程

15.2 SPIKE 101

15.2.1 模糊测试引擎

15.2.2 通用的基于行的TCP模糊器

15.3 基于块的协议建模

15.4 SPIKE的额外特性

15.4.1 特定于协议的模糊器

15.4.2 特定于协议的模糊测试脚本

15.4.3 通用的基于脚本的模糊器

15.5 编写SPIKE NMAP模糊器脚本

15.6 小结

第16章 网络协议模糊测试：Windows平台上的自动化测试

16.1 ProtoFuzz的特性

16.1.1 包结构

16.1.2 捕获数据

16.1.3 解析数据

16.1.4 模糊变量

16.1.5 发送数据

16.2 必要的背景知识

16.2.1 错误检测

16.2.2 协议驱动程序

16.3 ProtoFuzz的开发

16.3.1 开发语言的选择

16.3.2 包捕获库

16.3.3 设计

16.4 实例研究

16.5 益处和改进的余地

16.6 小结

第17章 Web浏览器模糊测试

17.1 什么是Web浏览器模糊测试

17.2 目标

17.3 方法

17.3.1 测试方法

17.3.2 输入

17.4 漏洞

17.5 错误检测

17.6 小结

第18章 Web浏览器的模糊测试：自动化

18.1 组件对象模型的背景知识

18.1.1 在Nutshell中的发展历史

18.1.2 对象和接口

18.1.3 ActiveX

18.2 模糊器的开发

- 18.2.1枚举可加载的ActiveX控件
- 18.2.2属性、方法、参数和类型
- 18.2.3模糊测试和监视
- 18.3小结
- 第19章 内存数据的模糊测试
- 19.1内存数据模糊测试的概念及实施该测试的原因
- 19.2必需的背景知识
- 19.3究竟什么是内存数据模糊测试
- 19.4目标
- 19.5方法：变异循环插入
- 19.6方法：快照恢复变异
- 19.7测试速度和处理深度
- 19.8错误检测
- 19.9小结
- 第20章 内存数据的模糊测试：自动化
- 20.1所需要的特性集
- 20.2开发语言的选择
- 20.3Windows调试API
- 20.4将其整合在一起
- 20.4.1如何在特定点将“钩子”植入目标进程
- 20.4.2如何处理进程快照和恢复
- 20.4.3如何选择植入钩子的点
- 20.4.4如何对目标内存空间进行定位和变异
- 20.5一个最好的新工具PyDbg
- 20.6一个构想的示例
- 20.7小结
- 第三部分 高级模糊测试技术
- 第21章 模糊测试框架
- 21.1模糊测试框架的概念
- 21.2现有框架
- 21.2.1antiparser
- 21.2.2Dfuz
- 21.2.3SPIKE
- 21.2.4Peach
- 21.2.5通用模糊器
- 21.2.6Autodafe
- 21.3定制模糊器的实例研究：Shockwave Flash
- 21.3.1SWF文件的建模
- 21.3.2生成有效的数据
- 21.3.3对环境进行模糊测试
- 21.3.4测试方法
- 21.4模糊测试框架Sulley
- 21.4.1Sulley目录结构
- 21.4.2数据表示
- 21.4.3会话
- 21.4.4事后验证阶段
- 21.4.5一个完整的实例分析
- 21.5小结
- 第22章 自动化协议解析

22.1 模糊测试存在的问题是什么

22.2 启发式技术

22.2.1 代理模糊测试

22.2.2 改进的代理模糊测试

22.2.3 反汇编启发式规则

22.3 生物信息学

22.4 遗传算法

22.5 小结

第23章 模糊器跟踪

23.1 我们究竟想要跟踪什么

23.2 二进制代码可视化和基本块

23.2.1 CFG

23.2.2 CFG示例

23.3 构造一个模糊器跟踪器

23.3.1 刻画目标特征

23.3.2 跟踪

23.3.3 交叉引用

23.4 对一个代码覆盖工具的分析

23.4.1 PStalker设计概览

23.4.2 数据源

23.4.3 数据探查

23.4.4 数据捕获

23.4.5 局限性

23.4.6 数据存储

23.5 实例研究

23.5.1 测试策略

23.5.2 测试方法

23.6 益处和改进的余地

23.7 小结

第24章 智能故障检测

24.1 基本的错误检测方法

24.2 我们所要搜索的内容

24.3 选择模糊值时的注意事项

24.4 自动化的调试器监视

24.4.1 一个基本的调试器监视器

24.4.2 一个更加高级的调试器监视器

24.5 调试器在应用程序前看到的异常和调试器再次看到程序未捕获的异常的比较

24.6 动态二进制插装

24.7 小结

第四部分 展望

第25章 汲取的教训

25.1 软件开发生命周期

25.1.1 分析

25.1.2 设计

25.1.3 编码

25.1.4 测试

25.1.5 维护

25.1.6 在SDLC中实现模糊测试

25.2 开发者

25.3QA研究者

25.4安全问题研究者

25.5小结

第26章 展望

26.1商业工具

26.1.1安全性测试工具beSTORM

26.1.2BreakingPoint系统BPS-1000

26.1.3Codenomicon

26.1.4GLEG ProtoVer Professional

26.1.5安全性测试工具Mu-4000

26.1.6Security Innovation Holodeck

26.2发现漏洞的混合方法

26.3集成的测试平台

26.4小结

## 章节摘录

如果询问任何一位有成就的安全领域的研究者如何发现漏洞，很可能会得到一大堆答案。为什么？因为可用于安全性测试的方法太多，每种方法都有自己的优点和缺点。没有一种方法是绝对正确的，也没有一种方法能够揭示一个给定目标下所有可能的漏洞。在较高的层次上，有三种主要的方法用来发现安全漏洞：白盒测试、黑盒测试和灰盒测试。这些方法之间的差别是由测试者可得到的资源决定的。白盒测试是一个极端，它需要对所有资源进行充分地访问。这包括访问源代码、设计规约，甚至有可能还要访问程序员本人。黑盒测试是另一个极端，它几乎不需要知道软件的内部细节，很大程度上带有盲目测试的味道。远程Web应用的Pen测试是黑盒测试的一个好例子，它不需要访问程序源码。位于两个极端方法之间的是灰盒测试，它的定义因询问的人不同而不同。就我们的应用目的而言，灰盒测试需要访问编译后得到的二进制代码，或许还要访问一些基本的文档。本章将考察漏洞发掘的各种不同的高层和低层方法，起点是白盒测试，你以前可能听说过这种方法也被称为玻璃、透明或半透明测试。之后我们再定义黑盒测试和灰盒测试，模糊测试可能就属于后两者。我们将阐述每种方法的利弊，这些方法的利弊将成为本书后面介绍模糊测试时所需要的背景知识。模糊测试只是漏洞发掘的一种方法，了解其他可选的实用方法也是相当重要的。

# 《模糊测试》

## 编辑推荐

《模糊测试强制性安全漏洞发掘》可作为开发者，安全工程师、测试人员以及QA专业人员的参考用书。掌握揭露安全性缺陷的最强大技术模糊测试现在已经发展成为一种最有效的软件安全性测试方法。模糊测试是指将一个随机的数据源作为程序的输入，然后系统地找出这些输入所引起的程序失效。著名的模糊测试专家将告诉你如何抢在别人之前使用模糊测试来揭示软件的弱点。《模糊测试强制性安全漏洞发掘》是第一部也是唯一一部自始至终讨论模糊测试的专著，将以往非正式的技巧转变为训练有素的最佳实践，进而将其总结为一种技术。作者首先回顾了模糊测试的工作原理并勾勒出模糊测试相比其他安全性测试方法的关键优势。然后，介绍了在查找网络协议，文件格式及Web应用安全漏洞中的先进的模糊测试，演示了自动模糊工具的用法，并给出多个说明模糊测试强大效力的历史案例。攻击者早已经开始使用模糊测试技术。当然，你也应该使用。不论你是一位开发者、一位安全工程师还是测试人员或QA专业人员，《模糊测试强制性安全漏洞发掘》都将教会你如何构建安全的软件系统。



# 《模糊测试》

## 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:[www.tushu111.com](http://www.tushu111.com)