

《恶意代码分析实战》

图书基本信息

书名：《恶意代码分析实战》

13位ISBN编号：9787121224682

出版时间：2014-4

作者：Michael Sikorski, Andrew Honig

页数：732

译者：诸葛建伟, 姜辉 张光凯

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.tushu000.com

《恶意代码分析实战》

内容概要

本书是一本内容全面的恶意代码分析技术指南，其内容兼顾理论，重在实践，从不同方面为读者讲解恶意代码分析的实用技术方法。

本书分为21章，覆盖恶意代码行为、恶意代码静态分析方法、恶意代码动态分析方法、恶意代码对抗与反对抗方法等，并包含了shellcode分析，C++恶意代码分析，以及64位恶意代码分析方法的介绍。本书多个章节后面都配有实验并配有实验的详细讲解与分析。通过每章的介绍及章后的实验，本书一步一个台阶地帮助初学者从零开始建立起恶意代码分析的基本技能。

本书获得业界的一致好评，IDA Pro的作者Ilfak Guilfanov这样评价本书：“一本恶意代码分析的实践入门指南，我把这本书推荐给所有希望解剖Windows恶意代码的读者”。

《恶意代码分析实战》

作者简介

本书作者Michael Sikorski, Andrew Honig虽不是安全领域中名声很大的人物，但从背景来看都是工作在一线的安全工程师与培训讲师，相信本书应是一本从事计算机病毒、恶意代码分析的专业人士，以及计算机病毒领域爱好者案头必备的一本实践指南参考书。

书籍目录

第0章 恶意代码分析技术入门1

- 0.1 恶意代码分析目标1
- 0.2 恶意代码分析技术2
 - 0.2.1 静态分析基础技术2
 - 0.2.2 动态分析基础技术2
 - 0.2.3 静态分析高级技术2
 - 0.2.4 动态分析高级技术2
- 0.3 恶意代码类型3
- 0.4 恶意代码分析通用规则4

第1篇 静态分析

第1章 静态分析基础技术 6

- 1.1 反病毒引擎扫描：实用的第一步6
- 1.2 哈希值：恶意代码的指纹7
- 1.3 查找字符串7
- 1.4 加壳与混淆恶意代码9
 - 1.4.1 文件加壳10
 - 1.4.2 使用PEiD 检测加壳10
- 1.5 PE 文件格式11
- 1.6 链接库与函数12
 - 1.6.1 静态链接、运行时链接与动态链接12
 - 1.6.2 使用Dependency Walker 工具探索动态链接函数13
 - 1.6.3 导入函数14
 - 1.6.4 导出函数15
- 1.7 静态分析技术实践15
 - 1.7.1 PotentialKeylogger.exe：一个未加壳的可执行文件15
 - 1.7.2 PackedProgram.exe：穷途末路18
- 1.8 PE 文件头与分节18
 - 1.8.1 使用PEview 来分析PE 文件19
 - 1.8.2 使用Resource Hacker 工具来查看资源节22
 - 1.8.3 使用其他的PE 文件工具23
 - 1.8.4 PE 文件头概述23
- 1.9 小结24
- 1.10 实验24

第2章 在虚拟机中分析恶意代码 27

- 2.1 虚拟机的结构27
- 2.2 创建恶意代码分析机28
 - 2.2.1 配置VMware29
 - 2.2.2 断开网络30
 - 2.2.3 创建主机模式网络30
 - 2.2.4 使用多个虚拟机30
- 2.3 使用恶意代码分析机31
 - 2.3.1 让恶意代码连接互联网31
 - 2.3.2 连接和断开外围设备32
 - 2.3.3 拍摄快照32
 - 2.3.4 从虚拟机传输文件33
- 2.4 使用VMware 进行恶意代码分析的风险34
- 2.5 记录/重放：重复计算机运行轨迹34

2.6 小结35

第3章 动态分析基础技术 36

3.1 沙箱：简便但粗糙的方法36

3.1.1 使用恶意代码沙箱36

3.1.2 沙箱的缺点37

3.2 运行恶意代码38

3.3 进程监视器39

3.3.1 进程监视器的显示40

3.3.2 进程监视器中的过滤41

3.4 使用进程浏览器（Process Explorer）来查看进程43

3.4.1 进程浏览器的显示43

3.4.2 使用验证选项44

3.4.3 比较字符串45

3.4.4 使用依赖遍历器（Dependency Walker）45

3.4.5 分析恶意文档46

3.5 使用Regshot 来比较注册表快照46

3.6 模拟网络47

3.6.1 使用ApateDNS47

3.6.2 使用Netcat 进行监视48

3.7 使用Wireshark 进行数据包监听49

3.8 使用INetSim51

3.9 基础动态分析工具实践52

3.10 小结55

3.11 实验56

第2篇 静态分析高级技术篇

第4章 x86 反汇编速成班 60

4.1 抽象层次60

4.2 逆向工程62

4.3 x86 体系结构62

4.3.1 内存63

4.3.2 指令64

4.3.3 操作码和字节序64

4.3.4 操作数65

4.3.5 寄存器65

4.3.6 简单指令67

4.3.7 栈70

4.3.8 条件指令73

4.3.9 分支指令73

4.3.10 重复指令74

4.3.11 C 语言主函数和偏移76

4.3.12 更多信息：Intel x86 Architecture Manual77

4.4 小结78

第5章 IDA Pro 79

5.1 加载一个可执行文件79

5.2 IDA Pro 接口81

5.2.1 反汇编窗口模式81

5.2.2 对分析有用的窗口83

5.2.3 返回到默认视图83

5.2.4 导航IDA Pro 83

- 5.2.5 搜索85
- 5.3 使用交叉引用86
 - 5.3.1 代码交叉引用87
 - 5.3.2 数据交叉引用88
- 5.4 分析函数88
- 5.5 使用图形选项89
- 5.6 增强反汇编91
 - 5.6.1 重命名位置91
 - 5.6.2 注释92
 - 5.6.3 格式化操作数92
 - 5.6.4 使用命名的常量93
 - 5.6.5 重新定义代码和数据94
- 5.7 用插件扩展IDA95
 - 5.7.1 使用IDC 脚本96
 - 5.7.2 使用IDAPython97
 - 5.7.3 使用商业插件97
- 5.8 小结98
- 5.9 实验98
- 第6章 识别汇编中的C代码结构 100
 - 6.1 全局与局部变量101
 - 6.2 反汇编算术操作102
 - 6.3 识别if 语句104
 - 6.3.1 用IDA Pro 图形化分析函数105
 - 6.3.2 识别嵌套的if 语句106
 - 6.4 识别循环107
 - 6.4.1 找到for 循环107
 - 6.4.2 找到while 循环109
 - 6.5 理解函数调用约定110
 - 6.5.1 cdecl110
 - 6.5.2 stdcall 111
 - 6.5.3 fastcall 111
 - 6.5.4 压栈与移动 111
 - 6.6 分析switch 语句112
 - 6.6.1 If 样式112
 - 6.6.2 跳转表114
 - 6.7 反汇编数组118
 - 6.8 识别结构体119
 - 6.9 分析链表遍历121
 - 6.10 小结123
 - 6.11 实验123
- 第7章 分析恶意Windows程序 126
 - 7.1 Windows API126
 - 7.1.1 类型和匈牙利表达法126
 - 7.1.2 句柄127
 - 7.1.3 文件系统函数127
 - 7.1.4 特殊文件128
 - 7.2 Windows 注册表129
 - 7.2.1 注册表根键130
 - 7.2.2 Regedit131

- 7.2.3 自启动程序131
- 7.2.4 常用注册表函数131
- 7.2.5 练习分析注册表操作代码132
- 7.2.6 使用.reg文件的注册表脚本133
- 7.3 网络API133
 - 7.3.1 伯克利兼容套接字134
 - 7.3.2 网络的服务器和客户端134
 - 7.3.3 WinINet API 135
- 7.4 跟踪恶意代码的运行136
 - 7.4.1 DLL136
 - 7.4.2 进程137
 - 7.4.3 线程139
 - 7.4.4 使用互斥量的进程间协作142
 - 7.4.5 服务143
 - 7.4.6 组件对象模型145
 - 7.4.7 异常：当事情出错时147
- 7.5 内核与用户模式148
- 7.6 原生API149
- 7.7 小结151
- 7.8 实验151
- 第3篇 动态分析高级技术篇
- 第8章 动态调试 154
 - 8.1 源代码级与汇编级的调试器154
 - 8.2 内核模式与用户模式调试155
 - 8.3 使用调试器155
 - 8.3.1 单步调试155
 - 8.3.2 单步跳过（Stepping-Over）和单步跳入（Stepping-Into）156
 - 8.3.3 用断点暂停执行157
 - 8.4 异常161
 - 8.4.1 首次和二次异常处理162
 - 8.4.2 常见异常162
 - 8.5 使用调试器修改可执行文件163
 - 8.6 修改可执行程序的实际163
 - 8.7 小结164
- 第9章 OllyDbg 165
 - 9.1 加载恶意代码165
 - 9.1.1 打开一个可执行文件165
 - 9.1.2 附加调试器到一个运行程序166
 - 9.2 OllyDbg 的接口167
 - 9.3 内存映射168
 - 9.3.1 基地址重定位169
 - 9.4 查看线程和堆栈170
 - 9.5 执行代码171
 - 9.6 断点172
 - 9.6.1 软件断点173
 - 9.6.2 条件断点174
 - 9.6.3 硬件断点175
 - 9.6.4 内存断点175
 - 9.7 加载DLL176

- 9.8 跟踪177
 - 9.8.1 标准回溯跟踪177
 - 9.8.2 堆栈调用跟踪178
 - 9.8.3 运行跟踪178
 - 9.8.4 跟踪Poison Ivy178
- 9.9 异常处理179
- 9.10 修补180
- 9.11 分析shellcode181
- 9.12 协助功能182
- 9.13 插件182
 - 9.13.1 OllyDump183
 - 9.13.2 调试器隐藏插件183
 - 9.13.3 命令行184
 - 9.13.4 书签185
- 9.14 脚本调试185
- 9.15 小结186
- 9.16 实验187
- 第10章 使用WinDbg 调试内核 189
 - 10.1 驱动与内核代码189
 - 10.2 安装内核调试191
 - 10.3 使用WinDbg193
 - 10.3.1 从内存中读取194
 - 10.3.2 使用算术操作符194
 - 10.3.3 设置断点194
 - 10.3.4 列举模块195
 - 10.4 微软符号表195
 - 10.4.1 搜索符号195
 - 10.4.2 查看结构信息196
 - 10.4.3 配置Windows 符号表198
 - 10.5 内核调试实践198
 - 10.5.1 用户空间的代码198
 - 10.5.2 内核模式的代码200
 - 10.5.3 查找驱动对象203
 - 10.6 Rootkit204
 - 10.6.1 Rootkit 分析实践205
 - 10.6.2 中断208
 - 10.7 加载驱动209
 - 10.8 Windows Vista、Windows 7 和x64 版本的内核问题209
 - 10.9 小结210
 - 10.10 实验210
- 第4篇 恶意代码功能篇
- 第11章 恶意代码行为 214
 - 11.1 下载器和启动器214
 - 11.2 后门 (backdoor) 214
 - 11.2.1 反向shell215
 - 11.2.2 远程控制工具216
 - 11.2.3 僵尸网络216
 - 11.2.4 远程控制工具与僵尸网络的比较217
 - 11.3 登录凭证窃密器217

- 11.3.1 GINA 拦截217
- 11.3.2 口令哈希转储218
- 11.3.3 击键记录221
- 11.4 存活机制223
 - 11.4.1 Windows 注册表223
 - 11.4.2 特洛伊木马化 (Trojanized) 系统二进制文件225
 - 11.4.3 DLL 加载顺序劫持227
- 11.5 提权228
 - 11.5.1 使用SeDebugPrivilege228
- 11.6 隐藏它的踪迹——用户态的Rootkit229
 - 11.6.1 IAT Hook 230
 - 11.6.2 Inline Hook 231
- 11.7 小结232
- 11.8 实验232
- 第12章 隐蔽的恶意代码启动 234
 - 12.1 启动器 (Launcher) 234
 - 12.2 进程注入234
 - 12.2.1 DLL 注入235
 - 12.2.2 直接注入237
 - 12.3 进程替换238
 - 12.4 钩子 (Hook) 注入240
 - 12.4.1 本地和远程钩子 (Hook) 240
 - 12.4.2 使用钩子的击键记录器241
 - 12.4.3 使用SetWindowsHookEx 241
 - 12.4.4 目标线程241
 - 12.5 Detours 242
 - 12.6 APC 注入243
 - 12.6.1 用户模式下APC 注入244
 - 12.6.2 内核模式的APC 注入245
 - 12.7 小结246
 - 12.8 实验246
- 第13章 数据加密 248
 - 13.1 分析加密算法的目的248
 - 13.2 简单的加密算法248
 - 13.2.1 凯撒密码249
 - 13.2.2 XOR249
 - 13.2.3 其他一些简单的加密策略254
 - 13.2.4 Base64255
 - 13.3 常见的加密算法258
 - 13.3.1 识别字符串和导入259
 - 13.3.2 查找加密常量259
 - 13.3.3 查找高熵值内容261
 - 13.4 自定义加密262
 - 13.4.1 识别自定义加密263
 - 13.4.2 攻击者使用自定义加密的优势265
 - 13.5 解密265
 - 13.5.1 自解密265
 - 13.5.2 手动执行解密函数266
 - 13.5.3 使用通用的解密规范267

- 13.6 小结270
- 13.7 实验271
- 第14章 恶意代码的网络特征273
 - 14.1 网络应对措施273
 - 14.1.1 在原始环境中观察恶意代码273
 - 14.1.2 恶意行为的痕迹274
 - 14.1.3 OPSEC=操作安全性275
 - 14.2 安全地调查在线攻击者275
 - 14.2.1 间接性策略275
 - 14.2.2 获取IP 地址和域名信息276
 - 14.3 基于内容的网络应对措施278
 - 14.3.1 使用Snort 进行入侵检测278
 - 14.3.2 深入观察279
 - 14.4 结合动态和静态分析技术282
 - 14.4.1 过度分析的危险283
 - 14.4.2 在众目睽睽下隐藏283
 - 14.4.3 理解周边代码286
 - 14.4.4 寻找网络操作代码287
 - 14.4.5 了解网络内容的来源288
 - 14.4.6 硬编码数据 vs. 临时数据289
 - 14.4.7 确定和利用编码步骤289
 - 14.4.8 创建特征291
 - 14.4.9 分析解析例程292
 - 14.4.10 针对多个元素294
 - 14.5 了解攻击者的意图295
 - 14.6 小结296
 - 14.7 实验296
- 第5篇 逆向工程
- 第15章 对抗反汇编 300
 - 15.1 何谓对抗反汇编技术300
 - 15.2 挫败反汇编算法301
 - 15.2.1 线性反汇编302
 - 15.2.2 面向代码流的反汇编303
 - 15.3 对抗反汇编技术306
 - 15.3.1 相同目标的跳转指令306
 - 15.3.2 固定条件的跳转指令307
 - 15.3.3 无效的反汇编指令308
 - 15.3.4 用IDA Pro 对指令进行NOP替换311
 - 15.4 混淆控制流图312
 - 15.4.1 函数指针问题312
 - 15.4.2 在IDA Pro 中添加代码的交叉引用313
 - 15.4.3 滥用返回指针313
 - 15.4.4 滥用结构化异常处理315
 - 15.5 挫败栈帧分析317
 - 15.6 小结320
 - 15.7 实验320
- 第16章 反调试技术322
 - 16.1 探测Windows 调试器322
 - 16.1.1 使用Windows API322

- 16.1.2 手动检测数据结构324
- 16.1.3 系统痕迹检测326
- 16.2 识别调试器的行为327
 - 16.2.1 INT 扫描327
 - 16.2.2 执行代码校验和检查328
 - 16.2.3 时钟检测328
- 16.3 干扰调试器的功能330
 - 16.3.1 使用TLS回调330
 - 16.3.2 使用异常332
 - 16.3.3 插入中断333
- 16.4 调试器漏洞334
 - 16.4.1 PE 头漏洞334
 - 16.4.2 OutputDebugString漏洞336
- 16.5 小结336
- 16.6 实验336
- 第17章 反虚拟机技术 338
 - 17.1 VMware 痕迹338
 - 17.1.1 绕过VMware 痕迹的探测340
 - 17.1.2 探测内存痕迹342
 - 17.2 查找漏洞指令342
 - 17.2.1 使用Red Pill 反虚拟机技术343
 - 17.2.2 使用No Pill 技术344
 - 17.2.3 查询I/O 通信端口344
 - 17.2.4 使用str 指令345
 - 17.2.5 反虚拟机的x86 指令346
 - 17.2.6 在IDA Pro 中高亮显示反虚拟机代码347
 - 17.2.7 使用ScoopyNG347
 - 17.3 调整设置348
 - 17.4 虚拟机逃逸349
 - 17.5 小结349
 - 17.6 实验349
- 第18章 加壳与脱壳 352
 - 18.1 剖析加壳352
 - 18.1.1 脱壳存根353
 - 18.1.2 加载可执行文件353
 - 18.1.3 解析导入函数表353
 - 18.1.4 尾部跳转354
 - 18.1.5 图示脱壳过程354
 - 18.2 识别加壳程序355
 - 18.2.1 加壳程序的标识355
 - 18.2.2 熵计算356
 - 18.3 脱壳选项356
 - 18.4 自动脱壳356
 - 18.5 手动脱壳357
 - 18.5.1 使用导入重构器重构导入表358
 - 18.5.2 查找OEP359
 - 18.5.3 手动修复导入表363
 - 18.6 常见壳的技巧与窍门364
 - 18.6.1 UPX 364

18.6.2 PECompact	365
18.6.3 ASPack	365
18.6.4 Petite	365
18.6.5 WinUpack	366
18.6.6 Themida	367
18.7 不完全脱壳情况下的分析	368
18.8 加壳DLL	368
18.9 小结	369
18.10 实验	369
第6篇 高级专题	
第19章 shellcode 分析	372
19.1 加载shellcode 进行分析	372
19.2 位置无关代码	373
19.3 识别执行位置	373
19.3.1 使用call/pop 指令	374
19.3.2 使用fnstenv 指令	376
19.4 手动符号解析	377
19.4.1 在内存中找到kernel32.dll	378
19.4.2 解析PE 文件导出数据	380
19.4.3 使用散列过的导出符号名	382
19.5 一个完整的Hello World 例子	383
19.6 shellcode 编码	385
19.7 空指令雪橇	387
19.8 找到shellcode	387
19.9 小结	388
19.10 实验	389
第20章 C + + 代码分析	391
20.1 面向对象的编程语言	391
20.1.1 this 指针	392
20.1.2 重载与修饰	394
20.1.3 继承 (Inheritance) 和函数重写 (Overriding)	395
20.2 虚函数和非虚函数	396
20.2.1 虚函数表的使用	398
20.2.2 识别虚函数表	399
20.3 创建和销毁对象	400
20.4 小结	401
20.5 实验	401
第21章 64 位恶意代码	403
21.1 为什么需要64 位恶意代码	403
21.2 x64 架构上的差别	404
21.2.1 x64 调用约定和栈使用上的差别	406
21.2.2 64 位异常处理	408
21.3 在Windows 64 位上的Windows 32 位	408
21.4 恶意代码功能上的64位提示	409
21.5 小结	410
21.6 实验	410
附录A 常见Windows 函数列表	412
附录B 流行的恶意代码分析工具列表	424
附录C 实验作业参考解答	435

附录D 致青春，基础软件开发的中國故事 691

附录E Syser 操作入门 695

《恶意代码分析实战》

精彩短评

- 1、内容广，适合上手入门。
- 2、这本书最大的亮点在于实战，每一章都有难度相当的上机实战问题，并指导你如何解决这些问题。
- 3、值得拥有。放在床头没事了翻翻挺不错，深度和广度都不错

1、随便在哪家网上书城进行搜索可以知道，在计算机安全类，特别是恶意代码分析领域的书籍可谓凤毛麟角。如果哪位读者对于恶意代码分析有浓厚的兴趣，要么是去一些大型的安全类论坛看他人的分析报告，要么是在众多的安全类书籍中，东找一点西凑一点地进行学习。这也就说明了市面上确实少有专门针对于恶意代码分析的，以实战为主的书籍。而对于我这样的初学者而言，由于我希望成为一名反病毒工程师，因此我特别想能够阅读到在安全领域有着丰富经验的大牛的作品。而现实中的大牛往往很忙，大部分也不愿意将自己的经验记录下来。于是这就出现了学习的瓶颈，不知道如何学，也不知道在哪里能够系统地学。而由Michael Sikorski与Andrew Honig编写的《恶意代码分析实战》这本书，正好填补了这片空白。在本书封底的显著位置，有这样一句话：恶意代码分析人员必须人手一册的经典！推销的噱头见多了，这本书是否真的如此重要呢？在我认真读完后的答案是——确实如此！首先本书的两位作者自身就拥有极其丰富的恶意代码分析经验，而本书正是全面总结了二位作者在实际分析过程中所遇到的恶意程序的各种特性。他们将恶意程序的种种行为总结得非常全面，从某种意义上来说，本书就是一部反恶意程序的百科全书，这是现在市面上任何教程都无法比拟的。书中不仅仅告诉了读者恶意程序的实现原理，更能教会我们应当如何去应对。由于我在阅读本书之前，也看过不少安全类的教程与书籍，并且在恶意代码分析方面多多少少懂得一些，因此我就很难以“小白”的眼光来看待本书是否真的通俗易懂，只是以我目前的角度来说，其表述还是非常到位的，这就有助于我们快速入门。加上各章节之间并没有绝对的相关性，我们也完全可以挑选自己感兴趣的章节进行学习，而且如果以后遇到困难，也可以将本书作为辞典进行检索，毕竟目前我们能够见到的恶意程序的恶意行为，基本都包含在本书的范围之中了。阅读高手的思想，有助于我们更好地解决自身遇到的实际问题，从而提高我们自身的水平。当然，如果仅仅是“看”书的话，我个人认为，书中虽然涉猎广泛，但是深度略有不足，很多问题可以说就是一笔带过，这样一来，对自身实战水平的培养就略显不足。正所谓师父领进门修行在个人，幸好本书开创性地在每一章节的后面附上了一些练习，要注意，这些可不是普通的练习用的程序，而是实实在在的恶意程序，是需要虚拟机中进行运行分析的。所以我也希望各位读者切勿眼高手低，一定要踏踏实实地认真分析每一个课后练习，完成后再看书后答案，相信大家会收获颇丰。毕竟在计算机安全领域，空谈理论是没用的，最重要的还是动手实现的过程。由于本书课后的习题只能在作者的网站进行下载，境外的网站我们访问起来可能没那么方便，所以我就将这些习题下载并上传到了我的CSDN资源区中，网址如下

：http://download.csdn.net/detail/ioio_jy/8250331希望大家能够按照书中的方法，认真对待每一道题，做完之后再对照答案，从而真正将知识掌握在自己的手中。作为一本译著，尽管译者阵容强大，三位译者在安全领域也有着非常丰富的经验，但是书中的文字依旧摆脱不了绝大部分译著的通病，那就是过于追求原著的语言模式，从而使得很多语句非常生硬，很不符合我们中国人的语言习惯。要知道，英语语法和中文的语法在很多方面是截然不同的，如果硬要依照英文原著中的用词顺序进行翻译，势必会出现很多拗口的语句。而对于英文单词来说，如果我们查英汉辞典，其翻译可能只有几条，但是如果将这个单词运用于实际的句子，之后再翻译成中文，很多时候我们就会发现，如果严格依据辞典中的解释进行翻译，那么结果会很是奇怪。所以我觉得比较妥当的做法是，应当与时俱进，因地制宜地根据我们中文的特色与用词习惯进行翻译，对于某个单词的翻译，可能在辞典中并没有写明，甚至与原始解释多多少少还有些不同，但是放在中文里面却是合适的。我也翻译过书籍文章，因此我对此深有体会。其实我在翻译时，也难以跳出这个怪圈。我所见过的翻译得最为和谐的，是在《0day安全：软件漏洞分析技术（第2版）》中，作者对《Writing Small Shellcode》这篇文章的翻译。个人认为那位译者是彻底消化了那篇英文文章的精髓，并用自己的语言精确地转述出来了。我认为这就是翻译的最高境界，有兴趣的读者可以阅读一下。最后，对于本书，我是强力推荐的，也请大家支持正版书籍。

章节试读

1、《恶意代码分析实战》的笔记-第六章

这里可以结合另一本专门讲C语言代码的汇编结构的书一起看

2、《恶意代码分析实战》的笔记-第690页

700页的书竟然有将近300页是习题答案；翻译水平不太到位；夹带私货，介绍自己开发的产品；总体来看，差评。

《恶意代码分析实战》

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.tushu000.com