Broadview
www.broadview.com.cn

安全技术大系

Python灰帽子

——黑客与逆向工程师的Python编程之道

Gray Hat Python:
Python Programming for Hackers and Reverse Engineers

[美] Justin Seitz 著
丁赟卿 译
崔孝晨 审校

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

no starch press

推 荐 序

Python 是一款非常流行的脚本编程语言。特别是在黑客圈子里，你不会 Python 就几乎无法与国外的那些大牛们沟通。这一点我在 2008 年的 XCon，以及 2009 年的 idefense 高级逆向工程师培训中感触颇深。前一次是因为我落伍，几乎还不怎么会 Python，而后一次……记得当时我、海平和 Michael Ligh（他最近出版的 *Malware Analyst's Cookbook and DVD: Tools and Techniques for Fighting Malicious Code*）一书在 Amazon 上得了 7 颗五星！）讨论一些恶意软件分析技术时经常会用到 Python，从 Immunity Debugger 的 PyCommand、IDA 的 IDAPython 到纯用 Python 编写的 Volatility 工具（这是一款内存分析工具，用于发现 rootkit 之类的恶意软件）。Python 几乎无处不在！我也尝试过对 Volatility 进行了一些改进，在电子工业出版社举办的“在线安全”Open Party 上海站活动中，我以《利用内存分析的方法快速分析恶意软件》为题进行了演讲。

遗憾的是，之前市面上还没有一本关于如何利用黑客工具中提供的 Python（由于必须使用许多黑客工具中提供的库函数，所以这时你更像在用一种 Python 的方言编程）的书籍。故而，在进行相关编程时，我们总是要穿行于各种文档、资料之中，个中甘苦只自知。

本书的出版满足了这方面的需求，它会是我手头常备的一本书，啊不！是两本，一本备用，另一本因为经常翻看用不了多久就肯定会破烂不堪☺。

说到这本书的好处也许还不仅于此，它不仅是一本 Python 黑客编程方面的极佳参考书，同时也是一本软件调试和漏洞发掘方面很好的入门教材。这本书的作者从调试器的底层工作原理讲起，一路带你领略了 Python 在调试器、钩子、代码注入、fuzzing、反汇编器和模拟器中的应用，涵盖了软件调试和漏洞发掘中的各个方面，使你在循序渐进中了解这一研究领域目前最新研究成果的大略。

本书译者的翻译也很到位。不客气地说，不少好书是被糟糕的翻译耽误掉的。比如我在读大学时的一本中文版的参考书，我看了三遍没明白是怎么回事，后来想起老师推荐时用的是英文版，于是试着去图书馆借了本英文版，结果看一遍就明白了。不过这本书显然不属于此例。译者丁赟卿本来就是从事这一领域研究的，对原文意思的理解非常到位，中文用词也十分贴切。特别是这本书的英文版中原本是存在一些错误的（包括一些代码），译者在中文版中竟然已经一一予以纠正了，从这一点上也可以看出译者在翻译过程中的认真细致。

我已经啰啰嗦嗦地讲了不少了，你还在等什么？还不快去账台付钱？

崔孝晨

2010.12.16 于 Hannibal from Team509

译 者 序

早在逆向工程这一行当如今日一般具备诸多的工程要义之前，我们更多地使用“Crack”这一富有独立精神与草莽气息的词，来指代那些早期的为了反抗商业软件文化所固有的封闭特性与垄断本质而实施的破解行为。早期的 Cracker 们并不像如今的逆向工程师一般有福，有着大量强大丰富的自动化分析工具与成熟的方法论，然而在这些行业先行者身上所体现出的精神与文化值得我们学习。

Cracker 中的开山鼻祖+ORC 便是其中一位身兼工程师与哲学家气质理念的传奇人物，由其撰写的 18 篇《How to crack》作为最早的破解布道书广为流传，除了纯粹的技术内容外，其中不乏充满灵性的隐喻与哲学理念。在其笔下，令人望而生畏的反编译代码被喻为“代码丛林”（Code wood），而有幸踏入这块领地的 Cracker 们则有如苦行于山涧丛林的狩猎者，他们沿着目标所留下的蛛丝马迹苦苦追寻，在坚忍之中等待着正面相遇的那一刻因缘际会。然而完满福德与美好因缘并不会轻易眷顾身带凡夫之气的新手，在他们求得般若之前，也许免不了轮回于一次次触破水月镜花之后的悲喜，迷失于代码丛林时的自我怀疑与望眼欲穿，以及柳暗花明后的唏嘘短叹。Crack 的过程对于那些 Old School（守旧派）来说更像是一次精神修行，帮助你在这旅途中发现自我，找寻一个更好的自我。+ORC 在其撰写的教程中不止一次地提及修行的 Cracker 在 Crack 之时，不应忘记破解之禅（Zen of Crack）。

也许任何一门迷人醉心的文化或者技艺都逃不过在商业洪流磨砺下重建秩序的宿命，就像朋克文化与摇滚乐一样。随着各种黑帽，灰帽会议召开，越来越多的安全爱好者与黑客从地下浮出，试图在这个利益驱动的行业生态中寻求成功。于是在这个角色分工逐渐明细的行业中，黑帽们开发漏洞利用（exploit）与白帽们做应急响应的周期呈现交替缩短的态势。对于各个黑帽组织与安全机构的领导者而言，将一群天赋过人，却往往又性格迥异，而且成本不菲的安全工程师黏合到一起，在他们的创造性与工程可控性之间找到平衡点，成了他们需要解决的首要问题。谁能更好地处理这个问题，往往就能在这个分秒必争的博弈局面中抢得先机。

Python 语言似乎在这一衍变趋势与安全技术社区的共同诉求中成为了潮流方向，这一同时具备脚本语言简单、快捷与开发大型项目所需的严谨工程特性的精灵成为了众多黑客之间的揉合剂。关于 Python 社区中有一句广为流传的口号“生命短暂，请用 Python”，在 Python 身上所体现的理念与当今黑客与逆向工程师们所期待的不谋而合。这也许可以帮我们解释为何众多优秀的安全项目与黑客工具选择 Python 的原因。比如，目前在逆向工程行

业口碑甚佳的“白眉”便是一个使用纯 Python 实现的项目，白眉的作者同时也是业界的大牛 Pedram Amini 向来对 Python 偏爱有佳。另一款值得称道的调试器工具 Immunity Debugger 则是知名安全机构 Immunity Inc 的作品，来自 Immunity 的黑客们基于 Python 强大的底层操作能力与工程特性在繁杂琐细的操作系统底层与上层应用之间抽象出一层 API。从中我们可以领略到 Immunity Inc 的领导者，老牌黑客 Dave Aitel 在设计安全产品与协同众多安全研究者方面的智慧与卓越策略。这些非常值得安全技术研究与商业化发展不尽人意的国内机构借鉴与学习。

在本书的翻译过程中，我调试了书中所涉及的代码，发现了原书的一些问题，给 Justin 先生发了 E-mail，并得到了 Justin 先生的确认。

感谢 team 509 的 Hannibal 为本书担任审校一职，这是我完成翻译工作的信心来源。

感谢我的朋友赵文凯、宋超以及赵超的慷慨帮助。

感谢博文出版社的毕宁老师对于我初次翻译所犯错误的宽容与理解！

丁赟卿

2011 年 2 月于上海

前　言

“搞定了吗？”，这可能是在 Immunity 公司出现频率最高的一句话了。你也许会在类似以下的场景中听到这样的发问：“我正要给 Immunity Debugger 开发一个新的 ELF 加载器”，片刻停顿之后，“搞定了吗？”或者，“我刚发现了 IE 浏览器的一个 Bug!”又一片刻的沉寂之后，“那个漏洞利用程序搞定了吗？”在日常的安全项目中我们几乎无时无刻地须要创建或者改写自己的安全工具，并在这些频繁的活动中始终保持高速的开发节奏，这使得 Python 逐渐成为了这个舞台上的明星。你可以在下一个安全项目中选择 Python 作为自己的开发工具，也许你将会用它来创建一个特殊的反编译器或者开发一个完整的调试器。

当我走进位于南迈阿密海滩的 Ace Hardware（美国的一家连锁五金店），沿着摆放着螺丝刀的通道走过时，常常会感到目眩。你会看到接近 50 多种不同规格的螺丝刀以整齐的顺序陈列在货架上。每一种规格的螺丝刀都与紧邻的螺丝刀有着微小却又十分重要的区别。我不是一个合格的修理能手，因此无法准确地说出每一种螺丝刀最为理想的使用场合，但是我很确信类似的情况同样适用于我们的安全工具软件。尤其是当你在对 Web 类型或者其他类型的高度定制化的应用程序进行安全审计时，你会发现每一次的审计任务都会需要一把特殊的“螺丝刀”来解决问题。要知道能够及时地拼凑出一些类似 SQL API 函数钩子之类的安全小工具已经不止一次地拯救了 Immunity 的工作团队。当然这些工具并不仅仅适用于安全审计任务，一旦你能够使用钩子函数对 SQL API 进行拦截，你就可以轻易地编写出一个工具用于实时检测可疑的异常 SQL 查询，并及时向你的客户公司提供修复方案，以抵御那些来自顽固黑客们的攻击。

众所周知，要让你的每一个安全研究人员真正成为团队的一部分是一件棘手的事情。很多安全研究人员无论在面对何种类型的问题时，都怀揣着白手起家式的过度热情，企图将需要借助的工具库完全重写。比如说 Immunity 发现了某个 SSL Daemon 的一个安全漏洞，接下来很有可能发生的一件事就是，你突然发现你的某个安全研究人员居然正在试图从头开始编写一个 SSL 客户端。而他们对此通常给出的解释是“我能找到的 SSL 库都丑陋不堪”。

你需要尽力避免这种情况发生。事实情况并不是现有的 SSL 库丑陋不堪——它只是没有按照某个安全研究人员的特别偏好风格来设计而已。而我们真正需要做的是能够深入分

析大量的现有代码，快速地发现问题所在，并对其进行修改以适应自身所需，这才是及时地搭建出一个可用的 SSL 库，并用其开发出一个尚处于保鲜期内的漏洞利用程序的关键。而要做到这一点，你需要使你的安全研究员们能够像一个真正的团队一样去工作。一个熟练掌握了 Python 的安全研究人员就有了一个强大的武器，也许就像那些掌握了 Ruby 的安全研究人员一样。然而 Python 真正的与众不同之处显现在那些 Python 狂热分子们协同工作时，他们将犹如一个高速运转的超个体[①]一样战斗力惊人。正如你家厨房中的蚂蚁大军一样，当它们的数量足够组成一只大乌贼时，要杀死它们将比杀死一只乌贼棘手得多。而这正是本书极力告诉你的一个事实。

你也许已经为自己想做的事找到了一些工具。你也许会问："我已经有了一套 Visual Studio，里面附带了一个调试器，为什么还要去编写一个供自己专用的调试器。"或者"WinDbg 不是有一个插件接口了吗？"答案是肯定的。WinDbg 的确提供了插件接口，你可以通过那些 API 慢慢地拼凑出一些有用的东西。直到某一天你很可能又会说："Heck，如果我能和 5000 个 WinDbg 使用者互联该有多好啊，这样我们就可以互通各自的调试结果了"。如果你从一开始就选择了 Python，你只要写 100 行左右的代码就可以构建一个 XML-RPC 客户端与服务端，接下来整个团队可以同步地进行工作并使每个人及时地享有他人的成果和信息。

黑客绝不等同于逆向工程——你的目标并不是还原出整个应用程序的源码。你的目标是对软件系统获得比系统开发者自身更加深入的理解。一旦你能做到这一点，无论目标以何种形式出现，你将最终成功地渗透它，获得炙手可热的漏洞利用（exploit）。这也意味着你需要成为可视化、远程同步、图论、线性方程求解、静态分析技术以及其他很多方面的专家。因此，Immunity 决定将这些都标准化实现在 Python 平台上，这样一旦我们编写了一个图论算法，这个算法将在我们所有的工具中通用。

在第 6 章中，Justin 向你演示了如何使用一个钩子窃取 Firefox 浏览器中输入的用户名与密码。这正是一个恶意软件作者所做的事——从之前的一些相关报道中可以看出，恶意软件作者通常使用一些更为高级语言来编写此类程序（http://philosecurity.org/2009/01/12/interview-with-an-adware-author）。然而你同样可以使用 Python 在 15 分钟内编写出一个样例程序，用于向你的开发人员演示，让他们明白他们对自己的产品所做的安全假设并不成立。现在的一些软件公司出于他们所声称的安全考虑，在保护软件内部数据方面的投资花费不菲。而实际上他们所做的往往只是实现了一些版权保护和数字版权管理机制而已。

① 译者注：超个体就是那些只有依靠角色分工才能生存的群体，单独的个体将无法独立生存。蚂蚁社会就是一个典型的超个体，Dave Aitel 同学的思维很发散。

这正是本书试图教你的东西：快速创建安全工具的能力。你应当能够借助这种能力为你个人或者整个团队带来成功。而这也是安全工具开发的未来：快速实现、快速修改，以及快速互联。我想，最后你唯一剩下的问题也许就是：“搞定了吗？”

Immunity Ine 的创始人兼 CTO Dave Aitel
2009 年 2 月于美国佛罗里达州，迈阿密海滩

致　谢

我想借此机会感谢我的家人，对于他们在撰写本书过程中所表现出来的理解和支持。感谢我的四个可爱的孩子：Emily、Carter、Cohen 和 Brady，是你们给了爸爸完成此书的理由，我为拥有你们而感到无比幸福。我还要为我的姐姐和兄弟们在这个过程中所给予的鼓励说一声谢谢，你们自己都曾经历过著书立作的严苛和艰辛，拥有你们这些对技术作品出版感同身受的人真是受益匪浅——我爱你们。我还想对我的爸爸说，你的幽默感帮助我度过了那些难以执笔为继的日子——我爱你，老爸，不要停止让你周围的人发出笑声。

多亏了一路上众多优秀的安全研究人员的帮助才使得本书得以羽翼渐丰，他们是：Jared DeMott、Pedram Amini、Cody Pierce、Thomas Heller（传说中的无敌 Python 男）以及 Charlie Miller——我欠你们大伙一个大大的感谢。至于 Immunity 团队，毫无疑问，你们一直以来大度地支持着我来撰写此书，正是得益于你们的帮助，我不仅仅成长为一个 Python 小子，同时更成为了一名真正的开发人员和安全技术研究者。Nico 和 Dami，抽出了额外的时间来帮助我解决问题，对此表示不胜感激。Dave Aitel，我的技术编辑，始终驱使着本书的进度直至完成，并确保本书的逻辑性与可读性，在此致以莫大的感谢。对于另一个 Dave, Dave Falloon, 非常感谢你为我校阅此书，对于那些让我自己都哭笑不得的错误，对于你在 CanSecWest 大会上拯救了本人的笔记本电脑的英雄行径，以及你巫师一般神奇的网络知识，都令我印象深刻。

最后，是那些总是被放在最后感谢的家伙们——No Starch 出版团队。Tyler 与我经历了本书的整个出版过程（相信我，Tyler 将是你遇到的最有耐心的家伙），Bill 将鼓励声连同那个可爱的印有 Perl 小抄的咖啡杯赠予了我。Megan 在本书创作的尾声阶段为我减轻了众多的麻烦，还有其他为出版本书而工作在幕后的团队成员——谢谢你们！。我对你们为我所做的每一件事充满感激。现在这篇致谢词的篇幅快要跟格莱美的获奖感言有一拼了，最后再次说一声感谢给所有那些帮助过我，却可能被我忘记提及的朋友们——你们清楚自己之于本书的意义。

Justin Seitz

简　介

我为了进行黑客技术研究而特地学习了 Python 这门语言，我敢断言在这个领域中的众多其他同行们也是如此。我曾经花费了大量的时间来寻找一种能够同时适用于黑客技术和逆向工程领域的编程语言，就在几年前，Python 成为了黑客编程领域内显而易见的王者。而一个不尽人如意的事实是，到目前为止还没有一本真正意义上的参考手册，来指导你将 Python 应用于不同的黑客技术场景中。你往往需要游走于各大论坛的技术讨论帖子中或者各种工具手册中。有时为了使你的工具能够正确地运转起来，花费一番不小的功夫来阅读和调试源代码也是司空见惯的情况。而本书正是致力于填补这方面的空缺，将引领你经历一次“旋风”之旅——你将看到 Python 这门语言是如何被应用在各式各样的黑客技术与逆向工程场景中的。

本书将向你揭示隐藏在各种黑客工具背后的原理机制，其中包括：调试器、后门技术、Fuzzer、仿真器以及代码注入技术，本书将向你一 一演示如何驾驭这些技术工具。除了学到如何使用现有的基于 Python 的工具之外，你还将学习如何使用 Python 构建自己的工具。需要有言在先的一点就是，这并不是一本大全式的参考手册！有大量使用 Python 编写的信息安全类工具未在此书中被提及。本书的信条是授之以渔，而非授之以鱼！你应当把从本书中所获得的技能灵活地应用于其他的场景中，根据自身的需求对你选择的其他 Python 工具进行调试，并做出扩展和定制。

阅读本书的方式不仅限于一种，如果你是个 Python 新手或者对于构建黑客工具尚感陌生，那么从前往后依次阅读对你来说是最好的选择，你将从最基本的理论开始，并在阅读本书的过程中编写相当数量的 Python 代码。当你阅读完本书时，你应当具备了自行解决各种黑客或逆向工程任务的能力。如果你对 Python 已有一定程度的了解，并且对 Ctype 库的使用驾轻就熟，那么不妨直接跳过第 1 章。对于那些行业浸沉已久的老手，相信你们可以在本书中来回穿梭自如，欢迎你们在日常工作中随时按需撷取本书中的代码片段或者相关章节。

本书在调试器相关的内容上花费了相当的篇幅，从第 2 章讲述调试器的基本原理开始，直至第 5 章介绍完 Immunity Debugger 为止。调试器对于任何一个真正的黑客而言都是至关重要的工具，因此我毫不吝惜笔墨来对它们进行广泛而全面的介绍。在之后的第 6 章和第 7 章中你将学到一些钩子和代码注入的技术，这些技术同样可以被调试器工具采用，作为控制程序流和操纵内存的手段。

本书接下来的焦点放在使用 Fuzzer 工具来攻破应用程序体系上。在第 8 章中，你将开

始学习基本的 Fuzzing 技术理论，我们将构建自己的文件 Fuzzing 工具。第 9 章将向你演示如何驾驭强大的 Fuzzing 框架——Sulley 来攻破一个现实世界中的 FTP daemon 程序。在第 10 章中，你将学习如何构建一个 Fuzzer 工具来攻击 Windows 驱动。

在第 11 章中，你将看到如何在 IDA Pro 中（一款流行的二进制静态分析工具）实现自动化执行静态分析任务。在第 12 章中，我们将介绍一款基于 Python 的仿真器——PyEmu，来为本书画上句号。

我试着使出现在本书中的代码尽量简洁，并在某些特定的地方加上了详细的注释以帮助你理解代码的本质。学习一门新的编程语言或者掌握一套陌生的函数库的过程少不了你自己的亲身实践，以及不断的自我纠正。我鼓励你以手动的方式将本书中的源程序键入电脑中！本书中出现的所有源码在 http://www.nostarch.com/ghpython.htm 恭候你的光临。

现在让我们开始编码吧！

Justin Seitz