HZ BOOKS
华章IT

no starch press

· 网络空间安全技术丛书 ·

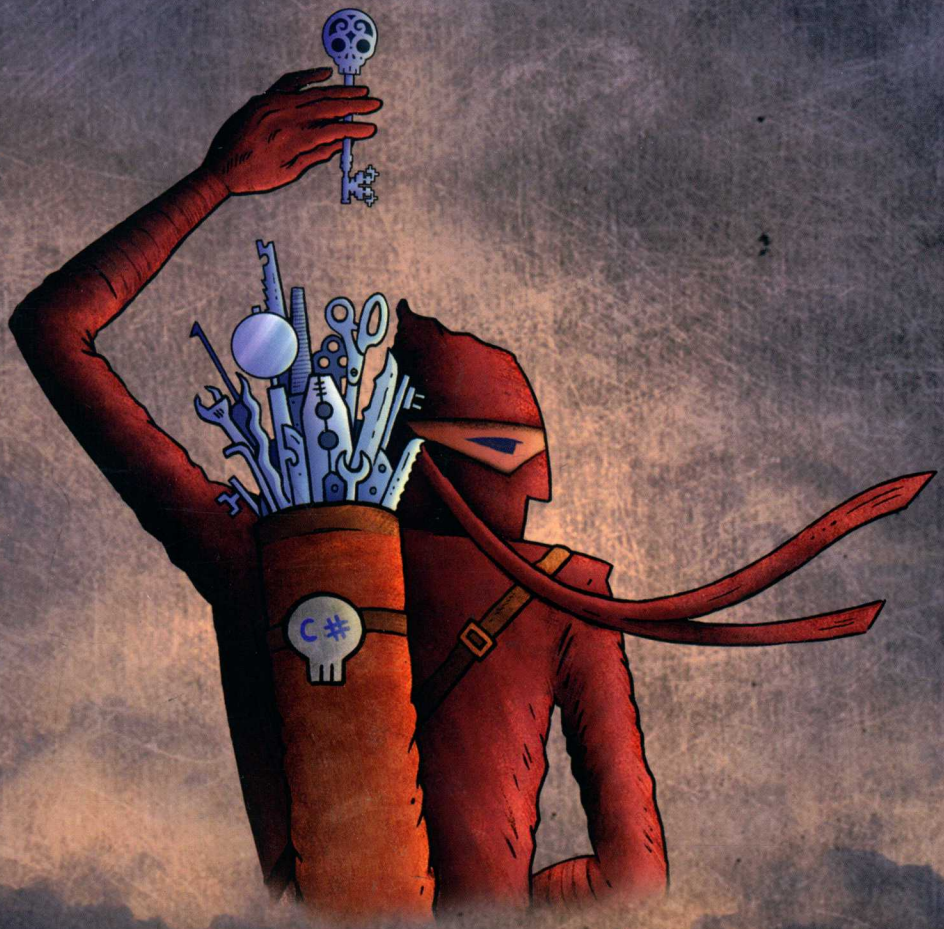
C#灰帽子

设计安全测试工具

GRAY HAT C#

A Hacker's Guide to Creating and Automating Security Tools

[美] 布兰德·佩里（Brandon Perry）著，王自亮 侯敬宜 李伟 译



机械工业出版社
China Machine Press

C#灰帽子

设计安全测试工具

GRAY HAT C#

A Hacker's Guide to Creating and Automating Security Tools

[美] 布兰德 · 佩里（ Brandon Perry ）著　王自亮 侯敬宜 李伟 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

C# 灰帽子：设计安全测试工具 /（美）布兰德·佩里（Brandon Perry）著；王自亮，侯敬宜，李伟译 . —北京：机械工业出版社，2018.2
（网络空间安全技术丛书）
书名原文：Gray Hat C#: A Hacker's Guide to Creating and Automating Security Tools

ISBN 978-7-111-59076-7

I. C… II. ①布… ②王… ③侯… ④李… III. C 语言－程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2018）第 025243 号

本书版权登记号：图字 01-2017-5817

C# 灰帽子：设计安全测试工具

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：陈佳媛　　责任校对：李秋荣
印　　刷：三河市宏图印务有限公司　　版　　次：2018 年 3 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：18.25
书　　号：ISBN 978-7-111-59076-7　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

译 者 序

互联网的快速发展给人们带来了快捷、高效的生产生活方式。随着互联网的加速渗透，网络已成为一个继海、陆、空、天之后与人类生活密切相关的第五空间，成为现代社会不可或缺的一部分。

各种各样丰富多彩的互联网应用在吸引大量用户的同时，也将自己暴露在了攻击者面前。震网病毒、棱镜门事件、Hacking Team 被黑事件、乌克兰电网系统遭攻击事件、希拉里邮件门、Mirai 病毒致使美国大规模断网事件……层出不穷的网络安全事件推动着网络安全从非主流走向主流，从附属变为有机组成部分，网络安全也成为整个安全体系的重要外延。特别是在《网络安全法》正式颁布实施之后，我国从法治的角度将网络安全的管理提升到一个新高度，对网络建设、运营、维护和使用的各方提出了具体要求。

少量设备的渗透测试或者安全防护所需的人力、物力投入都是可预期的。但如果设备数量达到一定的规模，任何组织和个人都需要面对量变引起质变所引发的一系列问题。如何省时省力而又高效地完成各项网络安全工作是每一位网络安全从业人员必须要解决的问题。

每一位需要渗透测试、风险评估的安全从业人员都有自己的“武器库”，有人擅长使用 Metasploit，有人喜欢使用 Nmap，也有人喜欢用 Nessus、OpenVAS。面对大量的结果数据，很多安全人员都有过“濒临崩溃”的经历，很多人都针对扫描、测试、分析等事项编写了自己的小工具，以求优化日常工作。只不过有的人喜欢用 Python，有的人喜欢用 C#。选择何种语言是个“仁者见仁智者见智”的事情，如果要展开讨论估计会争个面红耳赤，三天三夜也不会有结论。

C# 语言能够跻身常见编程语言之列，有许多先进的功能和特性，可以用来处理复杂的数据和应用。本书基于 C# 语言强大的核心库，略加改造，通过编程调用 Metasploit、

OpenVAS、Nessus 等渗透测试常见工具，来自动执行那些枯燥但又比较重要、基础的工作，如漏洞扫描、恶意软件分析以及事件响应。这样既能提升工作的趣味性，减少不必要的大力重复性工作，使得日常工作流程化、简单化，也切合了当前渗透测试、安全运营的 DevSecOps 趋势，有助于网络安全从业人员管理更为大型的网络，解决更多的安全问题。

如果你是一名希望从事网络安全工作的新手，那么可跟随本书的指导，更快地学到如何用 C# 来编程实现一些工具的优化甚至自动化；如果你是一名经验丰富的网络安全从业者，也可根据本书的提示，结合工作实战经验，编写出更满足自己需求的程序，让你的网络安全工作如虎添翼。

本书主要由王自亮、候敬宜、李伟完成翻译。我们力求做到在技术术语准确的前提下给读者带来最佳的阅读体验，但限于水平，难免有错误或疏漏，恳请广大读者朋友批评指正。

序

攻防双方在软件开发的过程中显然都需要决定哪种语言最适用。理想情况下一种语言不会仅仅简单地因为开发人员最喜欢而被选中。确定选择某种语言基于如下一系列问题：

- 我的主要目标执行环境是什么？
- 以这种语言编写的有效载荷的检测和记录状态是什么？
- 我的软件需要保持隐藏在什么级别（例如内存驻留）？
- 客户端和服务器端的支持情况如何？
- 是否有一个大的社区正在开发这门语言？
- 这种语言的学习曲线如何，可维护性怎样？

对这些问题 C# 有一些令人信服的答案。关于目标执行环境的问题，.NET 应该是在 Windows 环境下的最佳候选者，因为它已经和 Windows 打包在一起很多年了。但是随着 .NET 的开源，C# 现在成为可以在每种操作系统上运行的语言，自然 C# 应该是真正的跨平台语言。

C# 一直是 .NET 语言的通用语言。正如本书将会介绍的那样，由于其门槛低，开发人员众多，你将很快就能编写 C# 代码运行程序。此外，由于 .NET 是一种托管的类型丰富的语言，编译后的程序集可以简单地反编译为 C#。因此，编写攻击性 C# 代码不一定需要从零开始，而是可以从大量的 .NET 恶意软件样本中获取反编译的代码，阅读相应的源代码并“借用”它们的功能，甚至可以使用 .NET 反射 API 动态加载和执行现有的 .NET 恶意软件样本——当然，假设它们已经被逆向以确保不会做任何破坏。

有人花了很长时间将 PowerShell 引入主流市场，在 PowerShell 恶意软件激增之后，我的努力带来了大量的安全改进和日志功能。最新版本的 PowerShell（截至撰写本书时，最新版本为 v5）实现了比其他任何脚本语言更多的日志记录功能。从防守角度来看，这太棒了。从一个渗透测试者、红队成员，或对手的角度来看，这显著提高了攻

击者的门槛。对于一本关于 C# 的书，我为什么要提到这个？我花了多年的时间意识到，PowerShell 写得越多就越发现，攻击者通过在 C# 中而不是在 PowerShell 中开发工具，不会受到那么严格的限制，从而可以获得更高的灵活性。请允许我解释一下：

- .NET 提供了丰富的反射 API，允许用户轻松地在内存中加载和动态地与已编译的 C# 程序集进行交互。在 PowerShell 有效载荷上执行所有额外的检查后，反射 API 使攻击者可以通过开发仅用作 .NET 程序集加载器和运行器的 PowerShell 有效载荷以更好地躲避检测。
- 正如 Casey Smith（@subTee）所演示的那样，默认安装的 Windows 上有许多合法的 Microsoft 签名的可作为 C# 有效载荷的绝佳的宿主进程二进制文件—— msbuild.exe 是最隐蔽的宿主进程。使用 MSBuild 作为 C# 恶意软件的宿主进程完美体现了“不落地”的特点，即攻击者可以融入目标环境并占用最小的空间，且长时间驻留。
- 到目前为止，反病毒厂商仍不太了解运行时 .NET 程序集的功能。那里仍然有足够的非托管恶意代码，焦点还没有转移到有效地挂钩 .NET 运行时执行动态运行时检查。
- C# 有庞大的 .NET 类库，那些熟悉 PowerShell 的人将会发现向 C# 的过渡相对平滑，反过来，那些熟悉 C# 的人在将其技能转移到其他 .NET 语言（如 PowerShell 和 F#）时的门槛更低。
- 与 PowerShell 一样，C# 也是一种高级语言，这意味着开发人员不必关心底层编码工作和内存管理范例，但是，有时候需要底层编码（例如，与 Win32 API 交互）。幸运的是，通过反射 API 和 P/Invoke 和封送处理接口，C# 可以根据需要获得底层编码能力。

每个人学习 C# 的动机不同。我的动机是需要扩展 PowerShell 技能以便在更多平台上更灵活地使用 .NET 代码。有的读者可能想扩充 C# 技能来获取攻击者的思维，有的读者可能希望将现有的攻击者思维应用于多种平台上。无论你的动机是什么，准备好通过本书来一次狂野之旅吧！本书作者为 C# 攻防工具开发提供了独特的经验和智慧。

Matt Graeber

Microsoft MVP

前　言

很多人问我为什么喜欢 C#。我原本是一个开源软件的支持者、忠实的 Linux 用户和 Metasploit 的贡献者（主要使用 Ruby 编写），然而我却把 C# 当作我最喜欢的语言，这似乎很奇怪。这是为什么呢？许多年前，当我开始使用 C# 的时候，Miguel de Icaza（因 GNOME 出名）开始了一个叫作 Mono 的小项目。在本质上，Mono 是一个 Microsoft .NET 框架的开源实现。C# 被提交为 ECMA 标准，微软将其吹捧为替代 Java 的框架（因为 C# 代码可以在一个系统或平台上编译并在其他地方运行），唯一的问题是微软只为 Windows 操作系统发布了 .NET 框架。Miguel 和一小群核心贡献者接受了使 Mono 项目成为 .NET 到达 Linux 社区的桥梁的重任。幸运的是，我的一个朋友建议我学习 C#，但是他也知道我对 Linux 很感兴趣，他为我指明了这个刚刚起步的项目的方向，看看我是否可以同时使用 C# 和 Linux。之后，我被 C# 深深吸引了。

C# 是一种优雅的语言，C# 的发明者和主要架构师 Anders Hejlsberg 曾经为 Pascal 编写编译器，然后为 Delphi 编写编译器，这些经历使他对各种编程语言的真正特点有深刻的理解。Hejlsberg 加入微软之后，于 2000 年左右推出了 C#。早年，C# 与 Java 共享了很多语言特性，比如 Java 的语法细节，但是随着时间的推移，C# 自成一派，并早于 Java 引入了一大堆功能，例如 LINQ、代理和匿名方法。使用 C#，你可以使用许多 C 和 C++ 的强大特性，可以使用 ASP.NET 栈或丰富的桌面应用程序编写完整的 Web 应用程序。在 Windows 上，WinForms 是 UI 库的首选，但对于 Linux 来说，GTK 和 QT 库更易于使用。最近，Mono 已经可以在 OS X 平台上支持 Cocoa 工具包，甚至支持 iPhone 和 Android。

为什么信任 Mono

贬低 Mono 项目和 C# 语言的人声称，Mono 等技术如果在非 Windows 的任何平台

上使用都是不安全的。他们认为微软将会停止开发 Mono，使 Mono 被遗忘到许多人都不会严肃谈论这个项目的程度。我不认为这是一个风险。在撰写本书时，微软不仅收购了 Xamarin 公司（该公司由 Miguel de Icaza 创建以支持 Mono 框架），而且微软拥有大量的开源的核心 .NET 框架。在 Steve Ballmer 的领导下，微软还以许多令人难以想象的方式接受了开源软件。新任首席执行官 Satya Nadella 表示，微软与开源软件对接根本没有任何问题，建议各种公司要积极参与 Mono 社区，以便使用微软的技术来进行移动开发。

本书的读者对象

在网络和应用安全工程师中，许多人在一定程度上依赖自动化地扫描漏洞或分析恶意软件。因为有很多安全专业人员喜欢使用各种操作系统，所以编写每个人都可以轻松运行的工具可能很困难。Mono 是一个不错的选择，因为它是跨平台的，并且有一个优秀的核心库集合，使安全专业人员将各种工作自动化变得简单。如果你有兴趣学习如何编写攻击性的 Exploit、自动扫描基础设施的漏洞、反编译其他 .NET 应用程序、读取离线注册表配置单元、创建自定义跨平台载荷，那么本书涵盖的许多内容都会让你快速入门（即使你没有 C# 的使用背景）。

本书的主要内容

在本书中，我们将介绍 C# 的基础知识，然后使用合适的、丰富的库快速实现实际能用的安全工具。在应用程序之外，我们会编写模糊工具来找到可能的漏洞，并编写代码对发现的任何漏洞进行全面利用。你将看到 C# 语言特性和核心库的强大功能。一旦学习了基础知识，我们将自动化目前流行的安全工具，比如 Nessus、Sqlmap 和 Cuckoo Sandbox。总之，在读完本书后，你将有一个包含库的执行方案列表，将许多安全专业人员经常执行的工作自动化。

第 1 章：C# 基础知识速成

在这一章中，我们通过简单的例子介绍 C# 面向对象编程的基础知识，但同时覆盖了各种各样的 C# 特性。我们从一个 Hello World 程序开始，然后构建小的类，以便更好地了解面向对象的概念，然后介绍更高级的 C# 特性，例如匿名方法和 P/Invoke。

第 2 章：模糊测试和漏洞利用技术

在这一章中，我们使用各种数据类型编写了一个寻找 XSS 和 SQL 注入的小型 HTTP 请求模糊工具（通过 HTTP 库与 Web 服务器通信）。

第 3 章：对 SOAP 终端进行模糊测试

在这一章中，我们采用前几章介绍的模糊测试工具概念，编写了另一个小型模糊测试工具，通过自动生成 HTTP 请求来检索和解析 SOAP WSDL，以查找潜在的 SQL 注入。

同时该章也会介绍如何从标准库中获得优秀 XML 库。

第 4 章：编写有效载荷

在这一章中，我们将重点放在 HTTP 上，继续编写有效载荷。我们首先创建几个简单的有效载荷——一个通过 TCP，另一个通过 UDP。然后学习如何在 Metasploit 中生成 x86/x86-64 shellcode 来创建跨平台和跨架构的有效载荷。

第 5 章：自动化运行 Nessus

在这一章中，为了将几个漏洞扫描程序自动化，我们回到 HTTP（第一个是 Nessus），通过编程了解如何创建、观察和报告 CIDR 扫描的范围。

第 6 章：自动化运行 Nexpose

在这一章中，我们继续专注于工具自动化，只不过转到 Nexpose 漏洞扫描器上。Nexpose 的 API 也是基于 HTTP 的，可以自动化扫描漏洞并创建报告。Rapid7 是 Nexpose 的创始人，为其社区产品提供一年免费的许可证，这对业余爱好者非常有用。

第 7 章：自动化运行 OpenVAS

在这一章中，我们专注于使用开源的 OpenVAS 使漏洞扫描自动化。OpenVAS 的 API 仅使用 TCP 套接字和 XML 实现通信协议，从根本上与 Nessus 和 Nexpose 不同。因为它也是免费的，所以对于希望通过有限的预算获得更多的漏洞扫描经验的爱好者来说很有用。

第 8 章：自动化运行 Cuckoo Sandbox

在这一章中，我们将使用 Cuckoo Sandbox 进行数字取证。使用易用的 REST JSON API 自动提交潜在的恶意软件样本，然后报告结果。

第 9 章：自动化运行 sqlmap

在这一章中，我们通过自动化执行 sqlmap，最大限度地发挥 SQL 注入的危害。首先

编写一些小工具，使用与 sqlmap 一起发送的易用的 JSON API 提交单个 URL。一旦熟悉了 sqlmap，我们会将其集成到第 3 章介绍的 SOAP WSDL 模糊测试工具中，自动利用和验证任何潜在的 SQL 注入漏洞。

第 10 章：自动化运行 ClamAV

在这一章中，我们开始关注与本机的非托管库进行交互。ClamAV 是一个受欢迎的开源反病毒项目，虽然不是用 .NET 语言编写的，但是我们仍然可以与其核心库以及 TCP 守护进程交互，这将允许远程使用。我们会在这两种情况下介绍如何将 ClamAV 自动化。

第 11 章：自动化运行 Metasploit

在这一章中，我们重点介绍 Metasploit，学习如何通过配有核心框架的 MSGPACK RPC 以编程的方式利用和报告植入 shell 的主机。

第 12 章：自动化运行 Arachni

在这一章中，我们关注通过双重许可配置黑盒 Web 应用程序扫描器 Arachni，这是一个免费开源的项目。使用更简单的 REST HTTP API 和随附项目更强大的 MSGPACK RPC，编写一些在扫描 URL 时自动报告调查结果的小工具。

第 13 章：反编译和逆向分析托管程序集

在这一章中，我们进行逆向工程。在 Windows 上有易于使用的 .NET 反编译器，但不适用于 Mac 或 Linux，所以我们自己写一个小的反编译器。

第 14 章：读取离线注册表项

在这一章中，我们通过检查二进制结构的 Windows 注册表，将注意力集中在注册表项上，从而转向事件响应。学习如何解析和读取离线注册表项，可以检索系统启动密码，它存储在注册表中，用于加密密码的散列。

致谢

可以说，写这本书花了 10 年时间！虽然在电脑上我只写了 3 年。我的家人和朋友肯定注意到我一直在不断地谈论 C#，但是他们非常宽容并理解我，成为我的超级耐心的听众。AHA 的兄弟姐妹们给了我这本书中许多项目的灵感，是本书的重要支柱。非常感谢 John Eldridge，一位亲密的朋友，是他带我进入 C# 世界，激发了我对编程的兴趣。Brian

Rogers 一直是我最好的技术资源之一，在本书的编写过程中，我们的思想产生了许多碰撞，并得到很多启发，他也是一个拥有敏锐眼光和见解的优秀的技术编辑。我的产品经理 Serena Yang 和 Alison Law 减轻了我反复修改书稿的痛苦。当然，Bill Pollock 和 Jan Cash 把我模糊的表述变成了每个人都可以读懂的清晰的句子。非常感谢 No Starch 的全体工作人员！

最后的说明

本书所介绍的内容远远不能反映 C# 的强大功能和构建自动化工具的潜力，特别是因为我们创建的许多库是灵活的、可扩展的。我希望这本书展示了将常用的或烦琐的任务自动化是多么简单，并鼓励你继续完善我们创造的工具。可以在 https://www.nostarch.com/ grayhatcsharp 找到本书的源代码和更新。

目　录