235章标红重点

第二章 攻击行径分析

攻击实施阶段的一般步骤

教材19页有叙述，比较详细

第三章 网络侦察

离线侦查手段-DNS

1.DNS泄露和DNS劫持的区别

- DNS泄露：是指未经授权地公开或泄露DNS记录的行为。这可能会导致攻击者获取关键信息，例如域名、IP地址和其他与域名相关的记录。攻击者可以利用这些信息进行针对性的攻击，如网络钓鱼和恶意软件分发。
- DNS劫持：DNS劫持是指黑客通过篡改DNS解析结果，将用户访问的网站域名指向恶意的IP地址或者伪装成合法的网站，从而实施网络攻击或者窃取用户信息。DNS劫持可能导致用户被重定向到恶意网站、无法访问正常网站、受到钓鱼攻击等。

2.防止DNS泄露/DNS劫持几种方案

- 建议使用复杂的密码重置路由器/软路由的默认密码。
- 最好远离不受信任的网站。
- 最好定期检查您的DNS设置是否已修改，并确保您的DNS服务器是安全的。
- 使用良好的安全软件和防病毒程序，并确保定期更新软件。
- 如果您已被感染，建议删除HOSTS文件的内容并重置Hosts File。
    - HOSTS文件路径（win11）：C:\Windows\System32\drivers\etc

- 重置方法：把HOSTS文件内容删除，然后把下面的内容复制粘贴进去，参考官方文档【点击进入】

3.DNS劫持（中毒）的工作流程

域名系统 (DNS) 是用于连接到 Internet 的计算机、服务或任何资源的分层命名系统。这些名称按域进行组织，以便在连接系统的丛林中轻松找到自己的出路。

DNS 层次结构通常如下所示：

根 → TLD（顶级域，如 .com 或 .org）→ 域（如 blogspot.com）→ IP 地址。

每当您在浏览器中输入 URL 并按 Enter 键时，互联网上的 DNS 服务器都会将该网站的名称转换为IP 地址，以便您的计算机可以与托管您要查找的网站的系统进行通信。这意味着 DNS 就像负责转换的翻译器，以便浏览器可以成功加载 Internet 资源。

但是，当 DNS 服务器本身受到攻击并提供虚假信息时会发生什么？浏览器可能会启动一个危险的网站，而不是合法网站。

这些协议是通过针对加载页面过程中的弱点而做出的，以将流量从合法 IP 地址重定向到非法 IP 地址。简单来说，黑客可以访问 DNS 服务器，然后根据自己的兴趣调整其目录。

因此，每当用户输入域名时，他们将被重定向到黑客设置的域，而不是他们想要访问的合法域。它给用户带来了重大风险。

4.DNS 中毒的危险

DNS中毒给用户带来了极大的风险，更危险的是用户根本不知道。下面列出了 DNS 中毒的一些危险。

恶意软件：当用户被重定向到虚假网站时，黑客就可以访问网络。黑客使用此访问权限在设备上安装恶意软件。

盗窃：黑客可以轻松通过 DNS 中毒从设备窃取个人数据。他们可以窃取财务凭证、登录凭证、安全号码和其他敏感数据等数据。

阻止设备安全更新：通过 DNS 中毒，黑客甚至可以阻止设备获取安全补丁更新。它可以帮助他们长期控制设备。

5.如何预防DNS中毒

防止 DNS 中毒对于保护您自己或您的公司免受网络犯罪分子的攻击至关重要。您可以应用多种方法来避免 DNS 中毒。

1、欺骗检测工具

DNS 欺骗检测工具会扫描发送给用户的 DNS 数据。它确保只向用户发送准确的 DNS 数据。

2、端到端加密

这是为确保从一端发送到另一端的 DNS 数据完全加密而采取的最常见的安全措施。加密可防止网络犯罪分子复制数据。

3、安装防火墙

安装支持在线病毒防护的防火墙或防病毒软件可以为您提供很多帮助。它将防止任何恶意软件或病毒进入您的设备，此外，它还可以清除您设备中已经存在的恶意软件。

随着网络技术的进步，世界也迎来了许多危险。使用互联网的风险与日俱增，越来越危险。因此，要减少或停止威胁，超级科技提醒大家都应该提高网络安全意识。

6.深入了解nslookup：域名查询与DNS解析

nslookup是一个命令行工具，用于查询Internet域名信息或诊断DNS服务器问题。它通过与DNS服务器交互，能够查询域名对应的IP地址、DNS记录的生存时间等信息，是网络管理员和开发人员的必备工具之一。

工作原理

nslookup通过发送DNS查询请求到DNS服务器，然后解析返回的响应来获取域名信息。它支持多种查询类型，如A记录、AAAA记录、MX记录等，可以用来查询不同类型的DNS记录。默认情况下，nslookup使用当前计算机所配置的DNS服务器进行查询。

7.IP与域名解析——nslookup与dig的功能和区别

nslookup命令用于**查询DNS服务器**，获取**特定主机名的IP**或**域名的MX记录**等。

dig是**domain information group**的简写，即**域名信息组**。

dig是一个常用的**域名查询工具**，可以从**DNS域名服务器**查询**主机地址信息**，获取到**详细的域名信息**。

区别：

dig会从**域名的官方DNS服务器上**获取到**精确的权威解答**；

nslookup只会得到**DNS解析服务器**保存在**Cache中的非权威解答。**

对于一些采用了**分布式服务器**和**CDN技术**的大型网站，用nslookup查询到的结果和dig查询结果往往不同。

离线探查手段——whois

1.深入理解Whois服务：查询域名注册信息的利器

在互联网世界中，域名是每个网站的身份标识。而Whois服务，就是用来查询域名注册信息的数据库。它是一个强大的工具，可以帮助我们了解域名的所有者、注册机构、注册日期等信息。那么，如何利用Whois服务查询域名注册信息呢？下面就让我们一起来了解。

一、Whois服务的**原理**

Whois服务是一个公开的数据库，其中包含了全球范围内的域名注册信息。当你输入一个域名，Whois服务会返回该域名的所有者、注册机构、注册日期、到期日期等信息。这些信息都是公开的，任何人都可以查询。

二、如何使用Whois服务查询域名注册信息

1. 在浏览器中输入域名想要查询一个域名的注册信息，你只需要在浏览器中输入域名，然后加上“/whois”后缀，例如：example.com/whois。
2. 查看域名注册信息浏览器会跳转到Whois服务的查询页面。在这个页面上，你可以看到该域名的所有者、注册机构、注册日期、到期日期等信息。这些信息都是以文本形式呈现的。三、常见问题及解决方法
3. 无法查询到域名注册信息如果你无法查询到某个域名的注册信息，可能是由于以下几个原因：域名未注册、查询方式不正确、Whois服务出现问题等。你可以尝试重新查询，或者使用其他可靠的Whois查询工具。
4. 查询结果不准确有时候，查询结果可能会出现误差。这可能是由于数据更新不及时、查询方式不正确等原因造成的。你可以尝试使用其他可靠的Whois查询工具，或者联系域名注册机构获取更准确的信息。四、如何保护自己的域名注册信息虽然Whois服务是一个强大的工具，但同时也暴露了域名所有者的敏感信息。如果你的域名包含个人或公司的敏感信息，你需要考虑使用隐私保护服务来隐藏你的个人信息。这样可以防止你的个人信息被滥用或受到不必要的攻击。

总结：通过以上内容，我们了解了Whois服务的原理以及如何使用它来查询域名注册信息。虽然Whois服务是一个公开的数据库，但它的强大之处在于可以帮助我们了解域名的所有者、注册机构等信息。同时，我们也需要注意保护自己的域名注册信息，避免个人信息被滥用或受到攻击。

2.什么是whois？

简单来说，whois就是一个用来查询域名是否已经被注册，以及注册域名的详细信息的数据库（如域名所有人、域名注册商、域名注册日期和过期日期等）。通过whois来实现对域名信息的查询。

早期的WHOIS查询多以命令列接口存在，但是现在出现了一些网页接口简化的线上查询工具，可以一次向不同的数据库查询。网页接口的查询工具仍然依赖WHOIS协议向服务器发送查询请求，命令列接口的工具仍然被系统管理员广泛使用。WHOIS通常使用TCP协议43埠。每个域名／IP的WHOIS信息由对应的管理机构保存。

不同域名后缀的whois信息需要到不同的whois数据库查询。如.com的whois数据库和.eu的就不同。目前国内提供WHOIS查询服务的网站有万网的 whois.hichina.com，Webmasterhome.cn，tool.chinaz.com， tool.admin5.com 等。

“WHOIS”是当前域名系统中不可或缺的一项信息服务。在使用域名进行Internet冲浪时，很多用户希望进一步了解域名、名字服务器的详细信息，这就会用到WHOIS。对于域名的注册服务机构（registrar）而言，要确认域名数据是否已经正确注册到域名注册中心（registry），也经常会用到WHOIS。直观来看，WHOIS就是链接到域名数据库的搜索引擎，一般来说是属于网络信息中心（NIC）所提供和维护的名字服务之一。

3.WHOIS系统组成

根据IETF标准 要求，WHOIS服务一般由WHOIS系统来提供。WHOIS系统是一个Client/Server系统。其中Client端主要负责：

1）提供访问WHOIS系统的用户接口；

2）生成查询并将其以适当的格式传送给Server；

3）接收Server传回的响应，并以用户可读的形式输出。

Server端则主要负责接收Client端的请求并发回响应数据。Internet上基于TCP协议的基本服务都有自己默认的TCP端口号，象HTTP服务的默认端口号为80，FTP服务的默认控制端口号为21（数据端口为20）等。同样作为Internet上核心服务之一的WHOIS服务，其Server端默认监听43号TCP端口，接收查询请求并产生响应。一般来说，Server端可以接收三种类型的信息查询：联系人、主机和域名。对于同一查询，Server端的输出应该具有一致性和稳定性。

4.WHOIS工作过程

WHOIS服务是一个在线的“请求/响应”式服务。WHOIS Server运行在后台监听43端口，当Internet用户搜索一个域名（或主机、联系人等其他信息）时，WHOIS Server首先建立一个与Client的TCP连接，然后接收用户请求的信息并据此查询后台域名数据库。如果

数据库中存在相应的记录，它会将相关信息如所有者、管理信息以及技术联络信息等，反馈给Client。待Server输出结束，Client关闭连接，至此，一个查询过程结束。

在线扫描——地址扫描

1.PING工作流程

ICMP应用实例——Ping

Ping 是 ICMP 的一个重要应用，主要用来测试两台主机之间的连通性。Ping 的原理是通过向目的主机发送 ICMP Echo 请求报文，目的主机收到之后会发送 Echo 回答报文。Ping 会根据时间和成功响应的次数估算出数据包往返时间以及丢包率。

Ping的完整工作流程：

Ping本质上是ICMP数据包，所以其工作流程就是ICMP数据包的发送与解析流程。

大致流程如下：

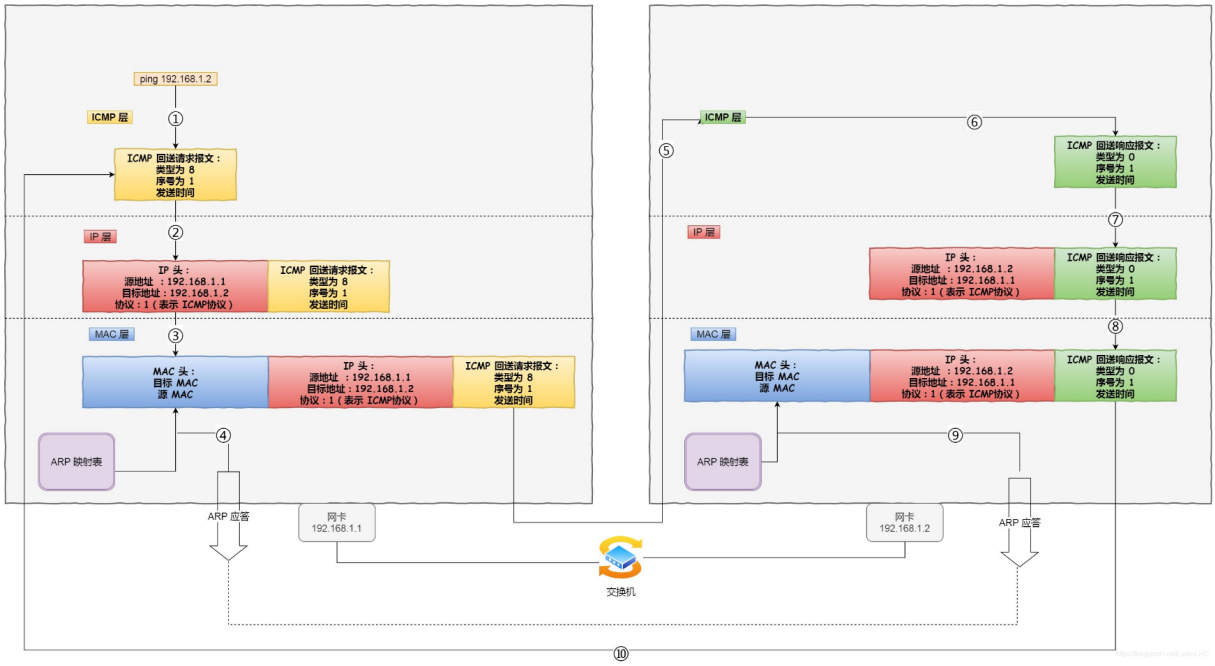
构造ICMP数据包→构造IP数据包→构造以太网数据帧----物理传输到目标主机-——→获取以太网数据帧→解析出IP数据包→解析出ICMP数据包→发送回送应答报文

本地主机处理流程：

1. 在本地，ping命令会构建一个ICMP数据包（构造回送请求报文）
2. 将该ICMP数据包和目标IP地址给IP协议，IP协议将本地地址作为源地址，与目的地址等构造IP数据包
3. 根据本地ARP缓存查找目的地址IP对应的MAC地址，如果缓存中没有则通过ARP协议找到对应IP的MAC地址。将MAC地址交给数据链路层以构造数据帧
4. 经物理层发送给目的主机

目的主机处理流程：

1. 目的主机接收到数据包
2. 物理层接收到二进制数据流经数据链路层，按照以太网协议解析出数据帧，如果数据帧中的目标MAC地址与本机MAC地址相同，则接收该数据包，否则丢弃该数据包。
3. 接收到该数据包之后，经网络层解析出IP数据包，通过IP包头中的协议字段判断出是ICMP数据包。之后解析出ICMP数据包，发现是回送请求报文（ping request）后马上构建一个ICMP回送应答报文（ping reply）
4. 将封装好的ICMP数据包经网络层、数据链路层、物理层发送回源主机



2.Traceroute工作原理

Traceroute 是 ICMP 的另一个应用，用来跟踪一个分组从源点到终点的路径。有2种实现方案：基于UDP实现和基于ICMP实现。

基于UDP实现traceroute工作原理

在基于UDP的实现中，客户端发送的数据包是通过UDP协议来传输的，使用了一个大于30000的端口号，服务器在收到这个数据包的时候会返回一个端口不可达的ICMP错误信息，客户端通过判断收到的错误信息是TTL超时还是端口不可达来判断数据包是否到达目标主机。流程如下：

1. 源主机向目的主机发送一连串的 IP 数据报（UDP报文）。第一个数据报 P1 的生存时间 TTL 设置为 1，当 P1 到达路径上的第一个路由器 R1 时，R1 收下它并把 TTL 减 1，此时 TTL 等于 0，R1 就把 P1 丢弃，并向源主机发送一个 ICMP 时间超过差错报告报文；
2. 源主机接着发送第二个数据报 P2，并把 TTL 设置为 2。P2 先到达 R1，R1 收下后把 TTL 减 1 再转发给 R2，R2 收下后也把 TTL 减 1，由于此时 TTL 等于 0，R2 就丢弃 P2，并向源主机发送一个 ICMP 时间超过差错报文。
3. 不断执行这样的步骤，直到最后一个数据报刚刚到达目的主机，主机不转发数据报，也不把 TTL 值减 1。但是因为数据报封装的是无法交付的 UDP，因此目的主机要向源主机发送 ICMP 终点不可达差错报告报文。

4. 之后源主机知道了到达目的主机所经过的路由器 IP 地址以及到达每个路由器的往返时间。

基于ICMP实现的tracert工作原理

在这一种实现中我们不使用UDP协议，而是直接发送一个ICMP回显请求（echo request）数据包，服务器在收到回显请求的时候会向客户端发送一个ICMP回显应答（echo reply）数据包。流程与上面相似，只是最后判断结束上为目标主机（而不是中间经过的主机或路由器）返回一个ICMP回显应答，则结束。

【版本2】

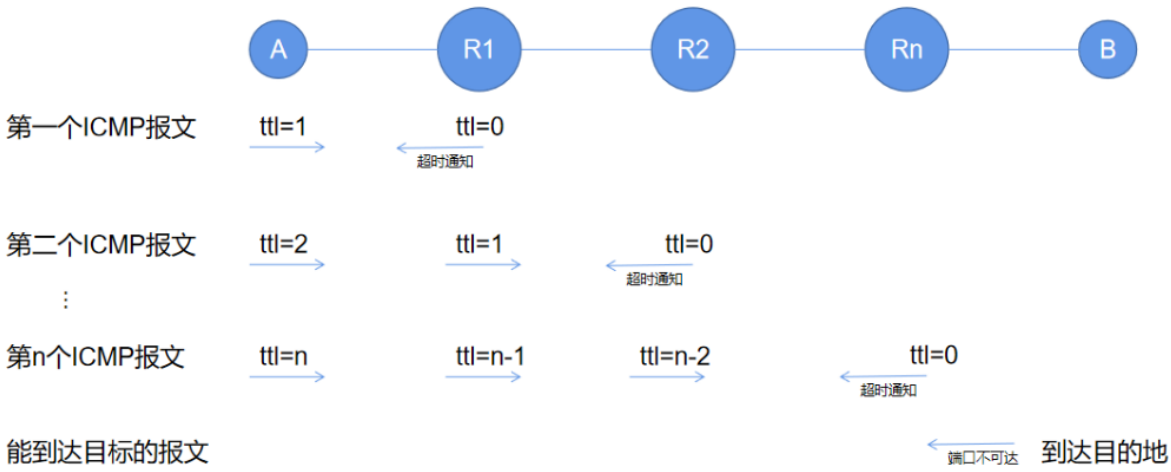
Tracert的工作原理：

Tracert 先发送 TTL 为 1 的回应数据包，并在随后的每次发送过程将 TTL 递增 1， 直到目标响应或 TTL 达到最大值，从而确定路由。TTL域每经过一个路由器转发后减1， TTL 减为 0 时，该路由器会将此数据报丢弃，并将超时回应数据报（包括数据报的源地址、内容和路由器的IP地址）发回源系统。

Tracert的工作过程：

1. 首先发送一个TTL为1的报文，当到达第一个路由器时TTL减1变为0，路由器就不再转发这个数据了，而直接丢弃，并且发送一个ICMP“超时”信息给源主机，并告知自己的IP地址；
2. 之后再发送一个TTL为2的报文，在第二跳返回TTL超时，这个过程不断进行，直至到达目的地；
3. 在目的地，由于数据报中使用了无效的端口号（缺省为33434），目的主机会返回一个ICMP目的地不可达消息，该tracert操作结束。

在上述过程中，tracert记录下每一个ICMP TTL超时消息的源地址，从而获得报文到达目的地所经过的网关的IP地址。



在线扫描——端口扫描

1.端口扫描技术

端口扫描技术

服务器上所开放的端口就是潜在的通信通道，也就是一个入侵通道。对目标计算机进行端口扫描，能得到许多有用的信息，进行端口扫描的方法很多，可以是手工进行扫描、也可以用端口扫描软件进行。扫描器通过选用远程TCP/IP不同的端口的服务，并记录目标给予的回答，通过这种方法可以搜集到很多关于目标主机的各种有用的信息，例如远程系统是否支持匿名登陆、是否存在可写的FTP目录、是否开放TELNET服务和HTTPD服务等。

扫描器能揭示一个网络的脆弱点。本身并不直接攻击网络漏洞，仅仅是发现目标机的弱点。漏洞扫描器探查远程服务器上可能存在的具有安全隐患的文件是否存在，它的socket建立过程和上面的端口扫描器是相同的，所不同的是漏洞扫描器通常使用80端口，然后对这个端口发送一个GET文件的请求，服务器接收到请求会返回文件内容，如果文件不存在则返回一个错误提示，通过接收返回内容可以判断文件是否存在。发送和接收数据需要使用函数send()和recv()，另外对流中存在的字符串进行判断需要使用函数strstr()，这除了需要具备socket函数库的知识以外，还需要一些有关string函数库的知识。

常用端口扫描技术

TCP connect()扫描

最基本的TCP扫描，操作系统提供的connect()系统调用可以用来与每一个感兴趣的目标计算机的端口进行连接。如果端口处于侦听状态，那么connect()就能成功。否则，这个端口是不能用的，即没有提供服务。这个技术的一个最大的优点是，你不需要任何权限。

系统中的任何用户都有权利使用这个调用。另一个好处就是速度，如果对每个目标端口以线性的方式，使用单独的connect()调用，那么将会花费相当长的时间，使用者可以通过同时打开多个套接字来加速扫描。使用非阻塞I/O允许你设置一个低的时间用尽周期，同时观察多个套接字。但这种方法的缺点是很容易被察觉，并且被防火墙将扫描信息包过滤掉。目标计算机的logs文件会显示一连串的连接和连接出错消息，并且能很快使它关闭。

TCP SYN扫描
通常认为是“半开放”扫描，这是因为扫描程序不必要打开一个完全的TCP连接。扫描程序发送的是一个SYN数据包，好象准备打开一个实际的连接并等待反应一样（参考TCP的三次握手建立一个TCP连接的过程）。一个SYN|ACK的返回信息表示端口处于侦听状态：返回RST表示端口没有处于侦听态。如果收到一个SYN|ACK，则扫描程序必须再发送一个RST信号，来关闭这个连接过程。这种扫描技术的优点在于一般不会在目标计算机上留下记录，但这种方法的缺点是必须要有root权限才能建立自己的SYN数据包。

TCP FIN 扫描
SYN扫描虽然是“半开放”方式扫描，但在某些时候也不能完全隐藏扫描者的动作，防火墙和包过滤器会对管理员指定的端口进行监视，有的程序能检测到这些扫描。相反，FIN数据包在扫描过程中却不会遇到过多问题，这种扫描方法的思想是关闭的端口会用适当的RST来回复FIN数据包。另一方面，打开的端口会忽略对FIN数据包的回复。这种方法和系统的实现有一定的关系，有的系统不管端口是否打开都会回复RST，在这种情况下此种扫描就不适用了。另外这种扫描方法可以非常容易的区分服务器是运行Unix系统还是NT系统。

IP段扫描
并不是新技术，它并不是直接发送TCP探测数据包，而是将数据包分成两个较小的IP段。这样就将一个TCP头分成好几个数据包，从而过滤器就很难探测到。但必须小心：一些程序在处理这些小数据包时会有些麻烦。

TCP 反向 ident扫描
ident 协议允许(rfc1413)看到通过TCP连接的任何进程的拥有者的用户名，即使这个连接不是由这个进程开始的。例如扫描者可以连接到http端口，然后用identd来发现服务器是否正在以root权限运行。这种方法只能在和目标端口建立了一个完整的TCP连接后才能看到。

FTP 返回攻击
FTP协议的一个有趣的特点是它支持代理（proxy）FTP连接，即入侵者可以从自己的计算机self.com和目标主机target.com的FTP server-PI(协议解释器)连接，建立一个控制通信连接。然后请求这个server-PI激活一个有效的server-DTP(数据传输进程)来给Internet上任何地方发送文件。对于一个User-DTP，尽管RFC明确地定义请求一个服务

器发送文件到另一个服务器是可以的，但现在这个方法并不是非常有效。这个协议的缺点是“能用来发送不能跟踪的邮件和新闻，给许多服务器造成打击，用尽磁盘，企图越过防火墙”。

UDP ICMP端口不能到达扫描
使用的是UDP协议，而非TCP/IP协议。由于UDP协议很简单，所以扫描变得相对比较困难。这是由于打开的端口对扫描探测并不发送确认信息，关闭的端口也并不需要发送一个错误数据包。幸运的是许多主机在向一个未打开的UDP端口发送数据包时，会返回一个ICMP_PORT_UNREACH错误，这样扫描者就能知道哪个端口是关闭的。UDP和ICMP错误都不保证能到达，因此这种扫描器必须还实现在一个包看上去是丢失的时候能重新传输。这种扫描方法是很慢的，因为RFC对ICMP错误消息的产生速率做了规定。同样这种扫描方法也需要具有root权限。

UDP recvfrom()和write() 扫描
当非root用户不能直接读到端口不能到达错误时，Linux能间接地在它们到达时通知用户。比如，对一个关闭的端口的第二个write()调用将失败。在非阻塞的UDP套接字上调用recvfrom()时，如果ICMP出错还没有到达时回返回EAGAIN-重试。如果ICMP到达时，返回ECONNREFUSED-连接被拒绝。这就是用来查看端口是否打开的技术。

服务器上所开放的端口就是潜在的通信通道，也就是一个入侵通道。对目标计算机进行端口扫描，能得到许多有用的信息，进行端口扫描的方法很多，可以是手工进行扫描、也可以用端口扫描软件进行。扫描器通过选用远程TCP/IP不同的端口的服务，并记录目标给予的回答，通过这种方法可以搜集到很多关于目标主机的各种有用的信息，例如远程系统是否支持匿名登陆、是否存在可写的FTP目录、是否开放TELNET服务和HTTPD服务等。

扫描器能揭示一个网络的脆弱点。本身并不直接攻击网络漏洞，仅仅是发现目标机的弱点。漏洞扫描器探查远程服务器上可能存在的具有安全隐患的文件是否存在，它的socket建立过程和上面的端口扫描器是相同的，所不同的是漏洞扫描器通常使用80端口，然后对这个端口发送一个GET文件的请求，服务器接收到请求会返回文件内容，如果文件不存在则返回一个错误提示，通过接收返回内容可以判断文件是否存在。发送和接收数据需要使用函数send()和recv()，另外对流中存在的字符串进行判断需要使用函数strstr()，这除了需要具备socket函数库的知识以外，还需要一些有关string函数库的知识。

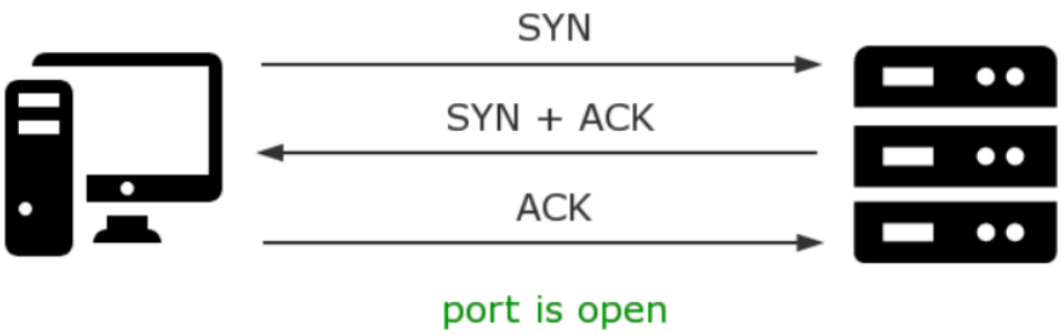
2.端口扫描原理

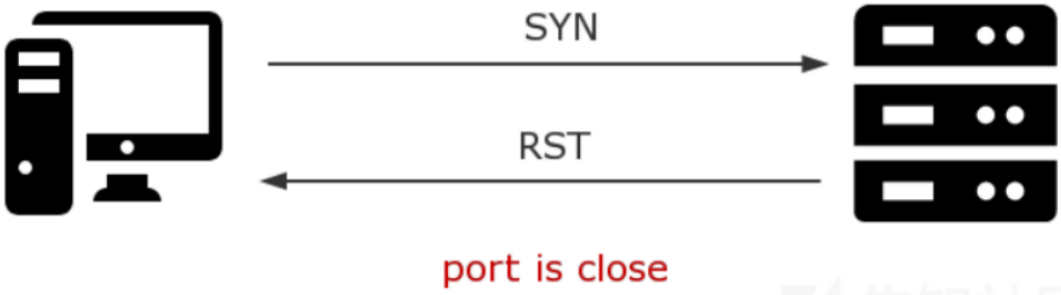
端口扫描，顾名思义，就是逐个对一段端口或指定的端口进行扫描。通过扫描结果可以知道一台计算机上都提供了哪些服务，然后就可以通过所提供的这些服务的已知漏洞就可进行攻击。其原理是当一个主机向远端一个服务器的某一个端口提出建立一个连接的请求，如果对方有此项服务，就会应答，如果对方未安装此项服务时，即使你向相应的端口发出请求，对方仍无应答，利用这个原理，如果对所有熟知端口或自己选定的某个范围内的熟知端口分别建立连接，并记录下远端服务器所给予的应答，通过查看一记录就可以知道目标服务器上都安装了哪些服务，这就是端口扫描，通过端口扫描，就可以搜集到很多关于目标主机的各种很有参考价值的信息。例如，对方是否提供FTP服务、WWW服务或其它服务。

常用端口扫描方法及原理

1、TCP CONNECT SCAN

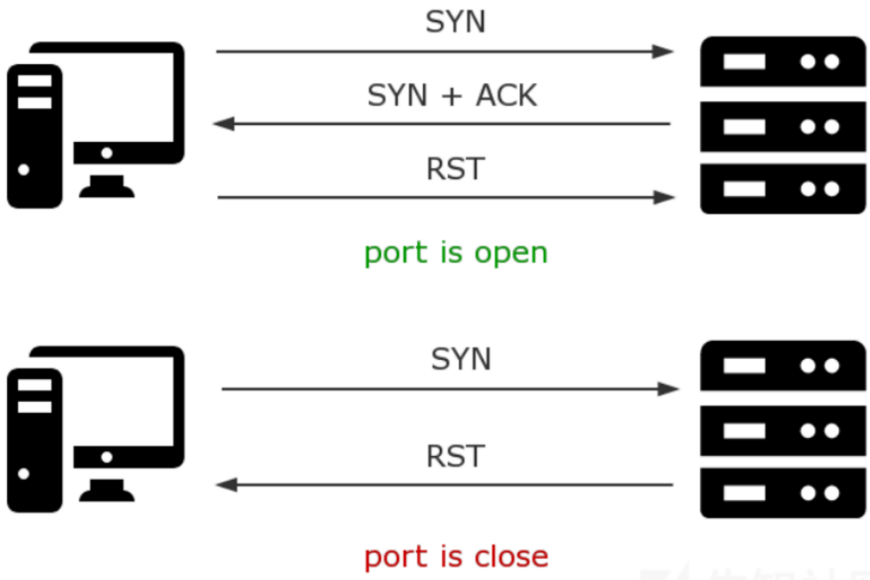
原理很简单，与目标端口建立3次握手，如果成功建立则为`open`，收到`RST`则为`close，此方法为全连接扫描。`





2、TCP SYN SCAN

也称为TCP半连接扫描，只发送三次握手的第一次SYN报文段，如果收到ACK+SYN则为open，收到RST则为close，这种好处是不必等待三次握手完全完成，速度快且不容易被防火墙记录进日志。



3、TCP FIN/Xmas Tree/NULL SCAN

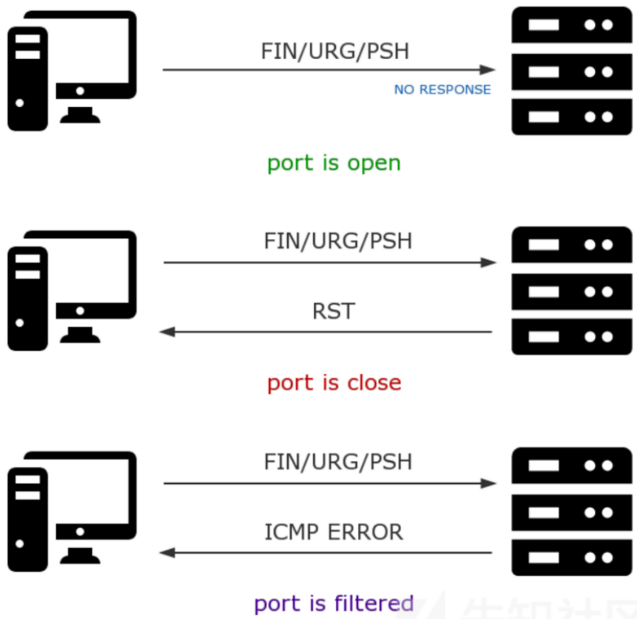
这几种探测方式原理相同，只是发送包的标志位不同。

FIN扫描：FIN标志位为1，其余标志位为0。

Xmas Tree扫描：也被称为Christmas Tree(圣诞树)扫描，它会设置FIN、URG和PSH标志位为1，其余标志为0。

NULL扫描：所有标志位都被设置为0。

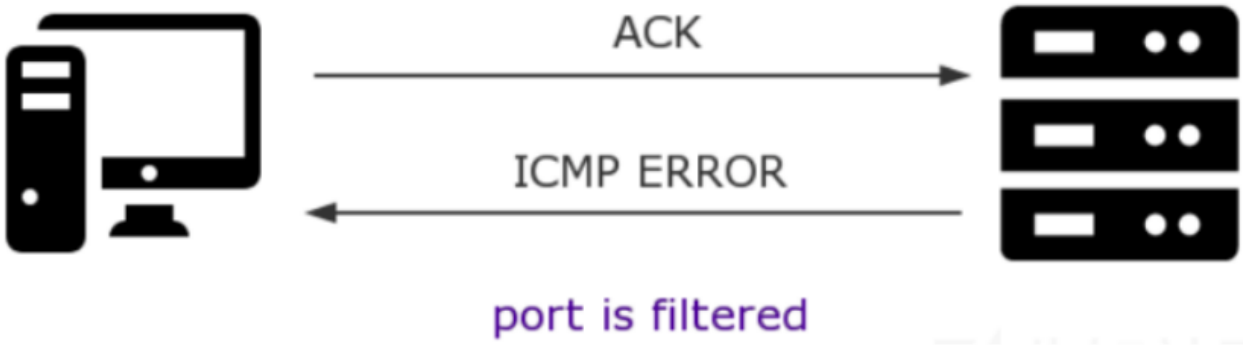
根据规定，理论上不会将不设置SYN，RST，或者ACK位的报文发送到开放端口，如果您确实收到了，丢弃该报文。当端口关闭时，任何不包含SYN，RST，或者ACK位的报文会导致一个RST返回。而当端口开放时，应该没有任何响应。



4、TCP ACK SCAN

当发送给对方一个含有 ACK 标志的TCP报文的时候，接收方会认为发生了错误(因为tcp建立连接时第一次握手报文不含ACK标志)，接收方会返回含有RST标志的报文，无论端口是开放或者关闭。所以，不能使用TCP ACK扫描来确定端口是否开放或者关闭。但是可以利用它来扫描防火墙的配置，用它来发现防火墙规则，确定它们是有状态的还是无状态的，哪些端口是被过滤的。

向服务端发送一个带有 ACK 标识的数据包，如果收到带有 RST 标识的响应，则说明服务端存在状态防火墙。当没有收到任何响应或者收到ICMP错误响应时表明该端口处于filtered状态。

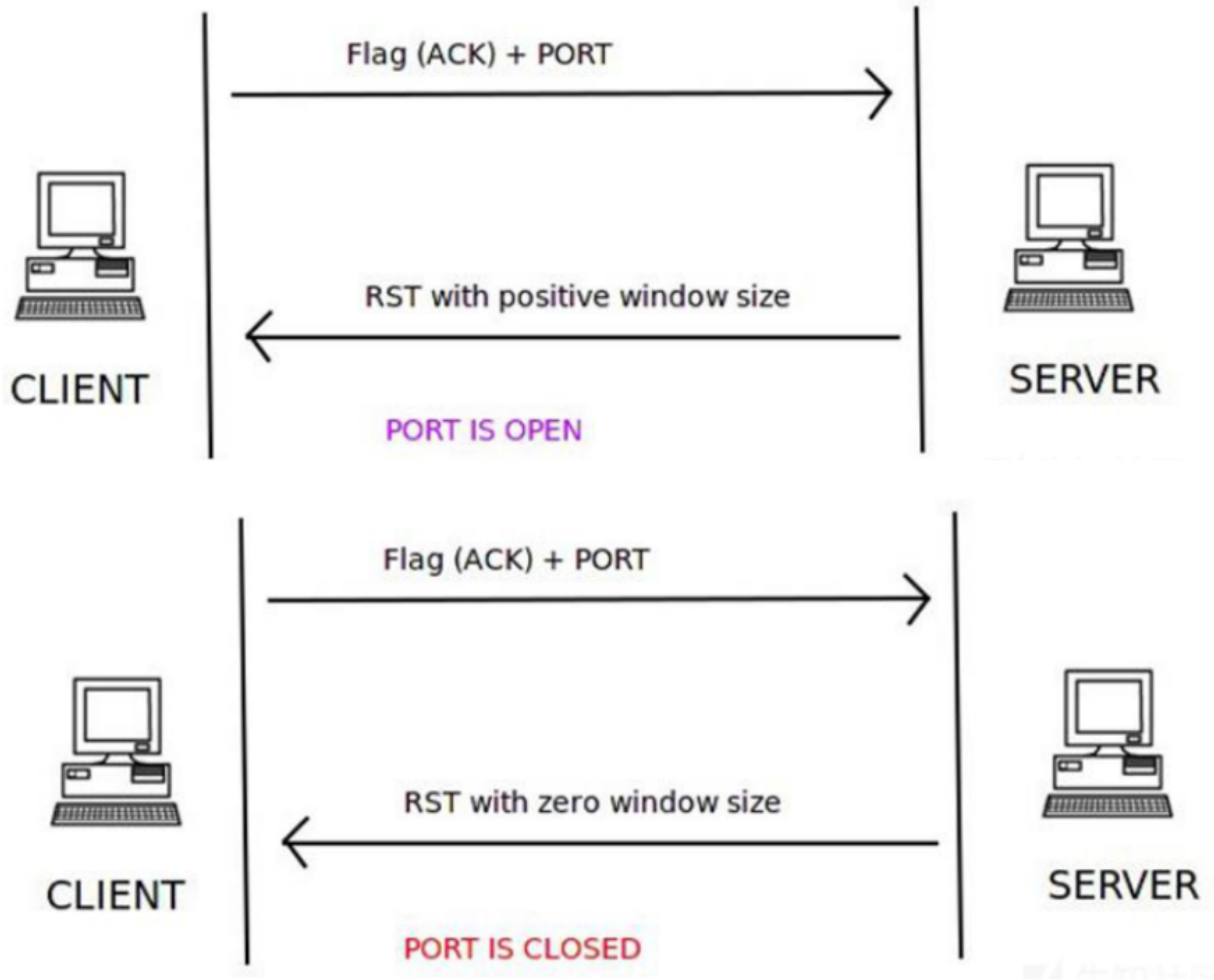


5、TCP WINDOW SCAN

TCP 窗口扫描的流程类似于 ACK 扫描，都是向服务端发送带有 ACK 标识的数据包，不同的在于 TCP 窗口扫描会检查收到的 RST 数据包中的窗口大小。

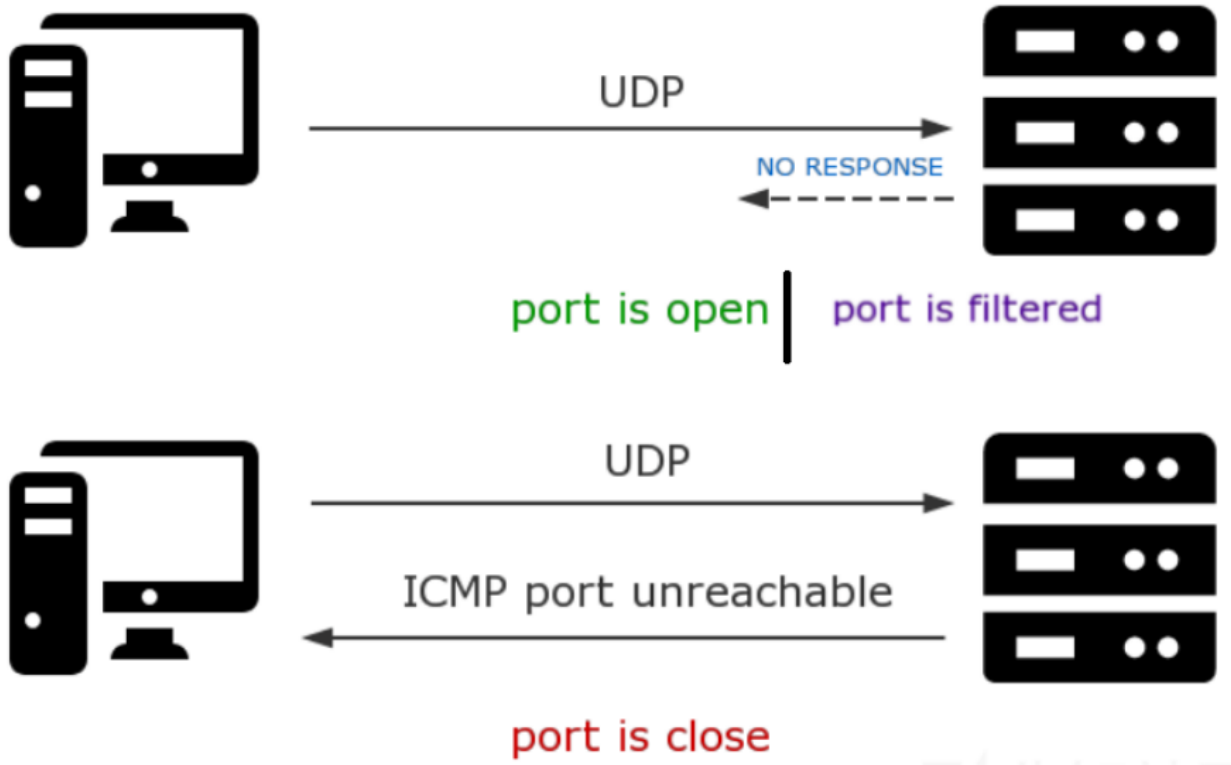
如果返回RST数据包说明没有防火墙过滤，端口开放或关闭根据RST包中的窗口大小判断。

如果 RST 数据包中的窗口大小不为零，则说明目标端口是开放的。如果 RST 数据包中的窗口大小为零，则说明目标端口处于关闭状态。



6、UDP SCAN

UDP扫描比较简单，一般如果返回ICMP port unreachable说明端口是关闭的，而如果没有回应或有回应(有些UDP服务是有回应的但不常见)则认为是open，但由于UDP的不可靠性，无法判断报文段是丢了还是没有回应，所以一般扫描器会发送多次，然后根据结果再判断。这也是为什么UDP扫描这么慢的原因。



虽然因特网上最流行的服务运行在TCP协议上，但UDP服务被广泛部署。 DNS，SNMP和DHCP（注册端口53,161 / 162和67/68）是最常见的三种。 由于UDP扫描通常比TCP更慢且更困难，因此某些安全审核员会忽略这些端口。 这是一个错误，因为可利用的UDP服务非常普遍，攻击者肯定不会忽略整个协议。

UDP扫描通过向每个目标端口发送UDP数据包来工作。 对于大多数端口，此数据包将为空（无有效负载），但对于一些更常见的端口，将发送特定于协议的有效负载。 根据响应或缺少响应，端口被分配给四种状态之一，如下表所示：

Probe Response	Assigned State	
Any UDP response from target port (unusual)	`open`	
No response received (even after retransmissions)	`open	filtered`
ICMP port unreachable error (type 3, code 3)	`closed`	
Other ICMP unreachable errors (type 3, code 1, 2, 9, 10, or 13)	`filtered`	

3.cjsPPT备注：

TCP的认证协议ident的漏洞：利用ident协议，可以获得TCP连接建立者的用户名。可以先连接到80端口，然后通过identd来发现服务器是否在root下运行．认证扫描(TCP ident扫描)：基于认证协议Ident的获得进程拥有者的反向扫描过程。认证协议(ident protocol)在RFC1413中的描述：每个类unix的操作系统都带有一个默认监听113端口的身份服务器。身份服务器能够提供本机某发起TCP连接进程的用户名。
FTP bounce attack 做法 在ftp协议中，数据连接可以与控制连接位于不同的机器上 让ftp server与目标主机建立连接，而且目标主机的端口可以指定 如果端口打开，则可以传输否则，返回"425 Can't build data connection: Connection refused.“ Ftp这个缺陷还可以被用来向目标(邮件,新闻)传送匿名信息

【交给GPT】

TCP Ident 协议的漏洞
漏洞描述：
Ident协议（身份认证协议）是一个基于TCP的协议，用于识别TCP连接的建立者。它定义在RFC 1413中。Ident协议的漏洞在于可以利用它来获取与服务器建立TCP连接的用户的用户名。

漏洞利用：
连接到服务器的80端口：攻击者可以首先与目标服务器的80端口（通常用于HTTP服务）建立连接。
查询身份服务器：然后，通过查询目标服务器上的Ident身份服务器（监听113端口），攻击者可以获取到该TCP连接的建立者的用户名。
发现root用户：如果返回的用户名是“root”，这意味着该服务是由超级用户（root）运行的，这对攻击者来说是有价值的信息，因为root用户权限非常高，意味着存在较大的安全风险。
防御措施：
关闭Ident服务（113端口）。
使用防火墙限制对113端口的访问。
使用非root用户运行网络服务。

FTP Bounce Attack
攻击描述：
FTP Bounce Attack利用了FTP协议中的一个缺陷。FTP协议允许数据连接与控制连接位于不同的机器上，这一特性被称为“第三方数据传输”（third-party data transfer）。这

种机制可以被恶意利用，使FTP服务器与攻击目标主机建立连接，并执行各种未经授权的操作。

攻击步骤：
与FTP服务器建立控制连接：攻击者首先与FTP服务器建立一个控制连接。
发送PORT命令：通过FTP协议的PORT命令，攻击者可以指示FTP服务器与第三方（目标主机）建立数据连接。
探测端口：攻击者可以通过FTP服务器尝试连接目标主机的不同端口。如果目标端口开放，FTP服务器将建立连接，否则返回错误信息（如“425 Can't build data connection: Connection refused.”）。
传输数据：利用FTP服务器开放的端口，攻击者可以向目标主机传输数据，发送匿名信息（如邮件、新闻）。
防御措施：
禁用FTP的第三方数据传输功能。
使用现代的FTP服务器软件，默认禁用或限制PORT命令的使用。
使用防火墙规则，限制FTP服务器与外部主机的连接行为。
采用SFTP（基于SSH的FTP）或FTPS（基于SSL/TLS的FTP）等更安全的传输协议。
总结
上述两种漏洞利用方法都涉及到通过TCP协议和特定服务端口的缺陷来获取敏感信息或进行未授权的操作。Ident协议的漏洞主要在于信息泄露，而FTP Bounce Attack利用了FTP协议的设计缺陷进行端口扫描和匿名数据传输。防御这些攻击的关键在于关闭或限制相应服务的使用，采用安全的配置和现代化的协议替代方案。

操作系统辨识

1.操作系统指纹识别概述

【概述】

操作系统指纹识别一般用来帮助用户识别某台设备上运行的操作系统类型。通过分析设备往网络发送的数据包中某些协议标记、选项和数据，我们可以推断发送这些数据包的操作系统。

只有确定了某台主机上运行的操作系统，攻击者才可以对目标机器发动相应的攻击。例如，如果要使用缓冲区溢出攻击，攻击者需要知道目标的确切操作系统与架构。

【为什么要进行操作系统指纹识别？】

能够探测到远程操作系统版本的网络侦察工具是非常有价值的，因为许多安全漏洞都依赖于操作系统的版本，如果没有这些信息，攻击者的攻击和利用都会受到限制。所以攻击发

动之前要做的就是收集目标操作系统信息。
例如，没有操作系统指纹识别技术，攻击者就无法知道目标服务器运行的是IIS服务还是Apache服务，有可能用IIS的漏洞去攻击Apache服务，最后无功而返。

【TCP指纹识别】

对操作系统的扫描是通过TCP/IP协议簇进行的。TCP/IP是互联网的基础协议，网络上所有的通信交互都通过该协议簇进行，因此操作系统必须实现该协议，使其与网络上其它计算机进行通信。IP用来将一个逻辑地址分配给网络上的机器，TCP用一种网络公认的方式传输IP数据包。这些标记对操作系统特别重要，每个操作系统根据数据包的不同类型做出不同的反应，如果是TCP包就发送到系统自己的网络栈。

TTL（Time to live）是由发送数据包的计算机或设备设置的。数据包每次经过路由转发后，该值都会被减1，所以如果一个数据包在网络上传输太久，途经太多跳数（机器间的路由），TTL值就会变为0（因为每经过一次路由转发都会将该值减1），该数据包就会被丢弃。

【ICMP指纹识别】

CMP协议也经常被用来进行指纹识别。许多traceroute功能使用ICMP协议发现起点到目标的网络路径。如果数据报没有被正确处理，不管是设备没有激活还是数据报自己的问题，ICMP都会返回错误消息，这些错误消息有时也很有用。

【操作系统指纹识别的类别】

1、主动指纹识别

主动指纹识别是指主动往远程主机发送数据包并对相应的响应进行分析的过程，使扫描器在更短的时间内获得比被动扫描更准确的结果。传统的方法是在探测到几个合法数据包时检查目标网络元素的TCP/IP栈行为。

（1）Nmap

网络侦察的第一步是确定网络中哪些机器是处于激活状态的。Nmap就是这样一款检测操作系统是流行工具，不仅可以检测远程操作系统是否运行，同时也可以执行各种端口扫描。

nmap通过向目标主机发送多个UDP与TCP数据包并分析其响应来进行操作系统指纹识别工作。在使用Nmap扫描系统的同时，该工具会根据响应包分析端口的打开和关闭状态。下图选项告诉Nmap在发现主机后不执行端口扫描，只打印出响应扫描器的主机，一般被称为“ping扫描”。

（2）Xprobe2

也可以使用xprobe2探测远程操作系统。Xprobe2根据Ofir Arkin的ICMP指纹识别研究执行远端TCP/IP协议栈的指纹识别工作，该工具依靠不同的方法识别操作系统是否处于激活状态，包括模糊签名匹配、概率猜测与联合匹配，以及一个签名数据库。

2、被动指纹识别

被动指纹识别是分析一台网络主机中发过来的数据包的过程。这种情况下，指纹识别工具被当作嗅探工具，不会向网络发送任何数据包。称其“被动”是因为这种方法不会与目标主机进行任何交互。基于这对这些数据包的嗅探跟踪，用户可以确定远程主机的操作系统。被动扫描通常比主动扫描更通用但准确性也更低，因为这种扫描对被分析数据的控制更少。

(1)NetworkMiner

NetworkMiner是一款网络取证分析工具。NetworkMiner也可被用作检测操作系统、会话、主机名、开放端口等的被动网络嗅探器和数据包捕获工具，而不需要向网络发送任何流量。

(2)用Ping命令检测操作系统

也可以通过执行ping命令，根据目标响应信息的TTL值来确定目标主机的操作系统类型。

Operating System	Time To Live	TCP Window Size
Linux (Kernel 2.4 and 2.6)	64	5840
Google Linux	64	5720
FreeBSD	64	65535
Windows XP	128	65535
Windows Vista and 7 (Server 2008)	128	8192
iOS 12.4 (Cisco Routers)	255	128

【预防】

完全屏蔽所有指纹识别攻击是个几乎不可能完成的任务，但我们可以通过一些措施加大攻击者的识别难度。例如我们必须确保外部主机无法直接扫描内部主机。

主动操作系统指纹识别一般也可以通过防火墙和主侵防御系统(IPS)解决。

banner抓取比较容易防御，可以通过对Apache的配置文件进行配置，限制其在头部列出的信息。

如果我们运行着某些服务并开放某些端口，可以屏蔽或删除触发某些错误时的显示的信息。

栈指纹技术

1.利用TCP/IP协议栈实现上的特点来辨识一个操作系统

利用TCP/IP协议栈实现上的特点来辨识一个操作系统（OS）的原理主要基于各操作系统在实现网络协议时存在的细微差异。以下是具体方法和所利用的特征：

1. TCP/IP堆栈实现的特性
   不同操作系统在实现TCP/IP协议时，采用了不同的默认参数和行为模式。通过分析这些特性，可以推断出目标系统的操作系统类型。

1.1 IP层特性
TTL（Time To Live）值：不同操作系统设定的默认TTL值不同。比如，Linux系统默认TTL值为64，Windows为128，UNIX系统通常为255。通过分析数据包的TTL值，可以初步判断出操作系统类型。

IP ID字段生成方式：不同操作系统生成IP ID字段的方式不同。有些操作系统采用顺序递增的方式，有些则采用随机生成。分析这些模式可以帮助识别操作系统。

1.2 TCP层特性
TCP初始序列号（ISN）生成方式：TCP连接建立时，初始序列号的生成方式在不同操作系统中有所不同。有些系统采用时间戳为基础的序列号生成方式，有些则采用完全随机的方式。

窗口大小：操作系统的默认TCP窗口大小不同。通过分析TCP数据包中的窗口大小，可以获得有关操作系统的信息。

TCP选项和参数：

MSS（最大段大小）：不同操作系统的MSS值不同。分析数据包中的MSS值可以帮助识别操作系统。
窗口缩放因子：这个TCP选项用于支持大窗口。不同操作系统支持不同的窗口缩放因子。
SACK（选择性确认）：有些操作系统默认启用SACK，而有些则不启用。
选项字段顺序：TCP头部中的选项字段顺序和内容在不同操作系统中会有所不同。分析这些顺序和内容可以帮助识别操作系统。

1. 协议实现的行为模式
   除了特定的字段和选项，不同操作系统在处理协议的方式上也有差异。以下是一些常

见的行为模式：

2.1 响应行为
SYN/ACK响应：在TCP三次握手过程中，向目标发送SYN包，分析其SYN/ACK响应的特性（如响应时间、窗口大小等）可以帮助识别操作系统。

异常包响应：发送畸形或异常的数据包，分析目标系统的响应行为。例如，向目标发送带有非法标志的TCP包，不同操作系统的响应会有所不同。

2.2 时间戳和重传
时间戳选项：有些操作系统在TCP包中包含时间戳选项，通过分析时间戳选项的值，可以推断出操作系统的类型。

重传行为：不同操作系统在处理重传时的行为也有所不同。例如，重传间隔时间和次数等。

2.NMAP栈指纹识别

nmap（Network Mapper）利用栈指纹识别技术来推断目标系统的操作系统。该技术的核心是分析目标系统在处理特定网络数据包时的行为特征，这些特征被称为“栈指纹”。以下是 Nmap 栈指纹识别的详细原理：

1. 探测数据包构造
   Nmap 发送一组精心设计的探测数据包到目标系统。这些数据包旨在触发目标系统的特定网络行为。常见的探测数据包类型包括：

TCP SYN 包：用于检测开放的端口并分析初始序列号生成。
TCP ACK 包：用于检查窗口大小和响应时间。
TCP FIN 包：用于检查目标系统对非法或不期望的 TCP 数据包的响应。
TCP Xmas 包：设置 FIN、URG 和 PSH 标志，用于检测对不常见数据包的响应。
TCP NULL 包：不设置任何标志，用于检查目标系统的响应行为。
ICMP Echo 请求：用于基本的连通性测试和分析 TTL 值。
UDP 包：用于检测开放的 UDP 端口和响应行为。
2. 响应捕获与分析
目标系统收到这些探测数据包后，会产生各种响应。Nmap 捕获这些响应并分析其各个字段和行为，包括：

IP 层字段：

TTL（Time To Live）：不同操作系统的默认 TTL 值不同。
IP ID：某些系统使用顺序递增的 IP ID，而其他系统使用随机值。
TCP 层字段：

初始序列号（ISN）：分析 ISN 生成模式。
窗口大小：不同操作系统的默认窗口大小不同。
TCP 选项顺序和内容：包括 MSS、窗口缩放、SACK 等选项的顺序和设置。
ICMP 响应：

Type 和 Code 字段：分析目标系统对 ICMP 请求的响应类型和代码。
数据字段：某些系统会在 ICMP 响应中包含特定的数据内容。
3. 特征匹配
Nmap 将收集到的响应特征与其内置的操作系统指纹数据库进行匹配。该数据库包含数千个操作系统和版本的指纹信息。具体步骤包括：

比较字段：将捕获的数据包字段与指纹数据库中的条目进行比对。
计算相似度：根据匹配的程度计算与各个指纹的相似度分数。
确定操作系统：选择相似度最高的指纹作为目标操作系统的识别结果。
4. 输出结果
Nmap 将识别结果显示给用户，通常包括以下信息：

可能的操作系统及其版本
识别的准确性（基于匹配相似度）
其他可能的操作系统选项（如果存在多个高相似度的匹配）

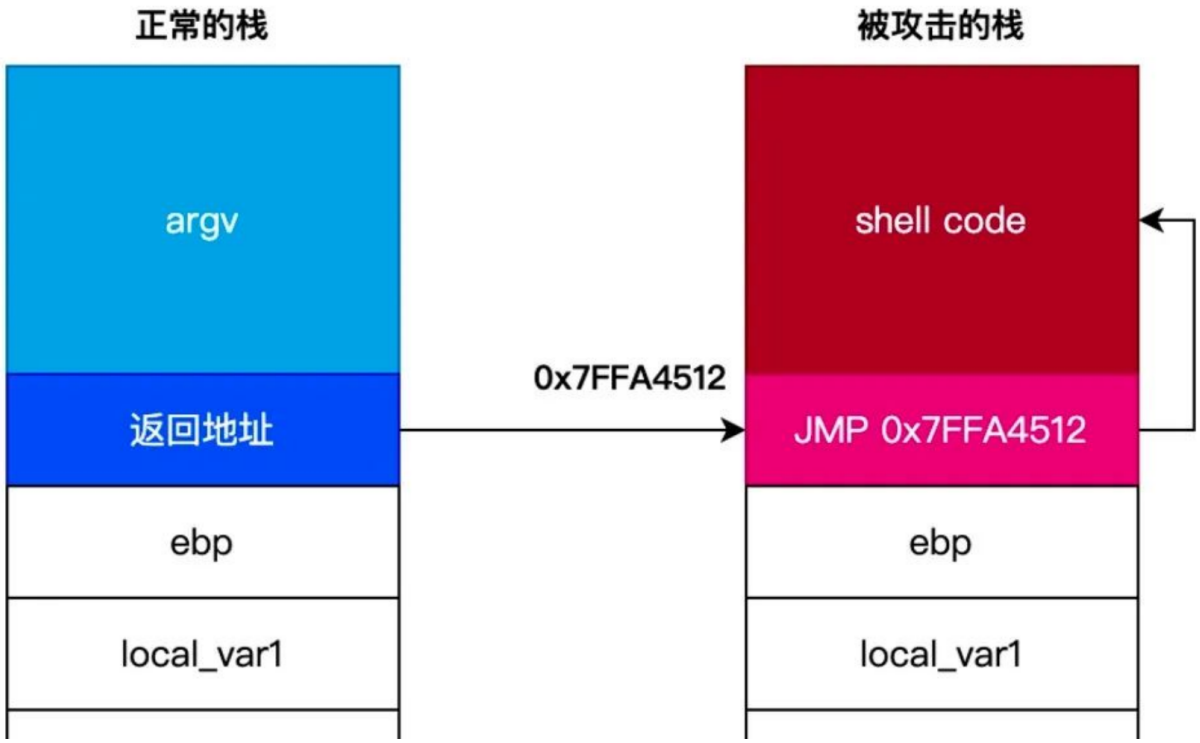
第五章

栈溢出攻击原理

1.攻击原理

栈溢出指的是程序向栈中某个变量中写入的字节数超过了这个变量本身所申请的字节数，因而导致与其相邻的栈中的变量的值被改变。这种问题是一种特定的缓冲区溢出漏洞，类似的还有堆溢出，bss 段溢出等溢出方式。栈溢出漏洞轻则可以使程序崩溃，重则可以使攻击者控制程序执行流程。此外，我们也不难发现，发生栈溢出的基本前提是：

程序必须向栈上写入数据。
写入的数据大小没有被良好地控制。



shell

Shell 是一个命令行解释器，它为用户提供了与操作系统进行交互的接口。它不仅可以解释和执行用户输入的命令，还可以进行编程和脚本编写。以下是 Shell 的功能和特点的详细介绍：

1. 命令行解释

Shell 主要功能之一是命令行解释。它接收用户输入的命令，并将其解析为操作系统可以理解的形式，然后执行这些命令。通过命令行解释，用户可以启动程序、管理文件和目录、检查系统状态等。

2. 使用保留字

Shell 使用了一些保留字来控制命令的流程和结构。这些保留字包括控制语句（如 `if`、`else`、`for`、`while` 等）和其他关键字（如 `function`、`case`、`select` 等）。保留字用于编写脚本和自动化任务，使 Shell 成为一个强大的编程环境。

3. 使用Shell元字符(通配符)

Shell 使用元字符（特殊字符）来实现通配符匹配和其他功能。常见的元字符包括：

- `*`：匹配零个或多个字符。例如，`*.txt` 匹配所有以 `.txt` 结尾的文件。
- `?`：匹配任意一个字符。例如，`file?.txt` 匹配 `file1.txt`、`fileA.txt` 等。
- `[]`：匹配括号内的任意一个字符。例如，`file[1-3].txt` 匹配 `file1.txt`、`file2.txt`、`file3.txt`。

4. 可处理程序命令

Shell 可以处理和执行各种程序命令。这些命令包括内置命令（如 `cd`、`echo`、`exit` 等）和外部命令（如 `ls`、`grep`、`awk` 等）。用户可以在 Shell 中组合使用这些命令来完成复杂的任务。

5. 使用输入输出重定向和管道

Shell 支持输入输出重定向和管道操作：

- **输入输出重定向**：将命令的输入或输出重定向到文件或其他命令。例如，`command > file` 将命令的输出重定向到文件，`command < file` 将文件的内容作为命令的输入。
- **管道**：使用管道符 `|` 将一个命令的输出作为下一个命令的输入。例如，`command1 | command2`。

6. 维护变量

Shell 维护环境变量和用户定义的变量。环境变量影响 Shell 的行为和操作环境，如 `PATH`、`HOME` 等。用户可以定义和使用自己的变量来存储和传递信息。例如：

```
MY_VAR="Hello, World"
echo $MY_VAR
```

7. 运行环境控制

Shell 可以控制和管理运行环境。用户可以通过 Shell 设置和修改环境变量，配置别名，控制进程的优先级和调度，管理后台和前台进程，控制终端会话等。

8. 支持Shell编程

Shell 提供了强大的编程功能，支持编写复杂的脚本和自动化任务。Shell 脚本可以包含变量、控制结构、函数、命令替换、循环和条件判断等编程元素。以下是一个简单的 Shell 脚本示例：

```
#!/bin/bash
# Simple script to greet the user
echo "Enter your name:"
read name
echo "Hello, $name!"
```

综述

Shell 的功能和特点使其成为一个强大而灵活的工具，适用于从简单命令执行到复杂脚本编写的各种任务。通过了解和利用这些功能，用户可以大大提高操作系统的使用效率和自动化程度。

栈溢出攻击步骤

栈溢出（Stack Overflow）攻击是一种常见的缓冲区溢出攻击，通过向程序的栈写入超出其分配范围的数据，覆盖返回地址或其他控制信息，从而改变程序的控制流，执行恶意代码。以下是栈溢出攻击的详细步骤：

1. 发现漏洞

- **描述**: 攻击者首先需要找到目标程序中的缓冲区溢出漏洞。这通常是由于程序在处理输入数据时没有进行适当的边界检查，导致缓冲区溢出。
- **示例**: 例如，一个使用 `gets()` 函数读取用户输入的程序，由于 `gets()` 不会检查输入长度，容易出现缓冲区溢出。

2. 了解程序结构

- **描述**: 攻击者需要了解目标程序的内存布局、函数调用栈和缓冲区位置。这通常通过逆向工程工具（如调试器或反汇编工具）来实现。
- **示例**: 使用 `gdb` 等调试工具分析程序的栈帧和内存布局。

3. 编写 Shell Code

Shell Code 是攻击者希望目标系统执行的恶意代码。它通常用汇编语言编写，然后转换为机器代码。

汇编语言程序

- **描述**: 编写 Shell Code 通常使用汇编语言，因为它允许精确控制指令和内存。
- **示例**: 一个简单的 Linux x86 下的 Shell Code，可以是执行 `/bin/sh` 的代码：

```
section .text
global _start

_start:
    xor eax, eax            ; 清除 eax 寄存器
    push eax                ; 压入空字符串到栈
    push 0x68732f2f         ; 压入字符串“//sh”
    push 0x6e69622f         ; 压入字符串“/bin”
    mov ebx, esp            ; 将栈指针赋值给 ebx
    push eax                ; 压入空字符串到栈
    push ebx                ; 压入 ebx 到栈
    mov ecx, esp            ; 将栈指针赋值给 ecx
    xor edx, edx            ; 清除 edx 寄存器
    mov al, 11              ; 将系统调用号 11 (execve) 赋值给 a
l
    int 0x80                ; 触发中断以执行系统调用
```

相对偏移

- **描述**: Shell Code 中的地址和偏移通常是相对的，以确保其在加载到内存的任何位置都可以正确执行。这是通过使用相对地址和寄存器进行地址计算实现的。
- **示例**: 使用 `ebx` 和 `ecx` 寄存器计算相对地址，确保 Shell Code 无需绝对地址即可运行。

消除 0

- **描述**: Shell Code 通常不能包含空字节（0x00），因为这些字节会被解释为字符串的结束符，导致 Shell Code 提前终止。为避免空字节，使用替代指令或对数据进行编码。
- **示例**: 使用 `xor eax, eax` 清除寄存器，而不是直接 `mov eax, 0`，因为后者可能会生成空字节。

机器代码

- **描述**: 汇编语言编写的 Shell Code 需要转换为机器代码（十六进制字节序列）以便嵌入到攻击载荷中。这可以通过汇编器（如 `nasm`）和转换工具（如 `objdump`）实现。
- **示例**: 使用 `nasm` 将汇编代码转换为机器代码：

```
nasm -f elf32 shellcode.asm
ld -o shellcode shellcode.o
objdump -d shellcode
```

4. NOP 填充和尝试地址

NOP 填充

- **描述**: NOP（No Operation）指令是一个无操作指令，用于填充缓冲区，以增加攻击成功的机会。NOP 指令使得 CPU 在遇到这些指令时什么也不做，而继续执行后续指令。
- **示例**: NOP 指令通常为 0x90（在 x86 架构下）。一个典型的攻击载荷会在 Shell Code 之前插入一段 NOP 滑道（NOP sled），如：

```
\\x90\\x90\\x90\\x90...（多次 NOP 指令）...Shell Code
```

尝试地址

- **描述**: 攻击者需要覆盖函数返回地址，使其指向 NOP 滑道或 Shell Code。由于内存地址可能会有所变化，攻击者通常会尝试多个地址，以增加成功概率。
- **示例**: 构建的攻击载荷可能如下：

```
Buffer + NOP sled + Shell Code + Padding + Return Address
（尝试多次）
```

5. 实施攻击

构建攻击载荷

- **描述**: 将超长输入数据构造成：填充数据 + NOP 滑道 + Shell Code + 填充数据 + 覆盖返回地址。
- **示例**:

```
AAAA...AAAA（64 bytes padding） + NOP sled + Shell Code + Return Address
```

发送攻击载荷

- **描述**: 使用合适的方法将攻击载荷注入目标程序的缓冲区。可以通过网络、文件输入或其他输入方式。
- **示例**: 通过网络请求或直接输入方式注入攻击载荷。

执行攻击

- **描述**: 当程序执行到被覆盖的返回地址时，控制流跳转到 NOP 滑道或 Shell Code，从而执行攻击者的代码。
- **示例**: 程序返回地址被覆盖后，执行到 Shell Code，打开一个 shell 或执行其他恶意操作。

总结

栈溢出攻击通过向程序的栈中写入过多数据覆盖返回地址，使得程序跳转到攻击者控制的 Shell Code。关键步骤包括编写和优化 Shell Code、构建攻击载荷以及利用 NOP 滑道增加攻击成功概率。了解这些步骤和原理对于理解和防御缓冲区溢出攻击至关重要。

UNIX与Windows

在进行栈溢出攻击时，不同的操作系统和体系结构会导致一些细节上的差异。以下是对上述文字的详细解释：

不同操作系统的内存布局

UNIX 系统

- **描述**: 在许多 UNIX 系统中，用户进程的虚拟地址空间通常是从 0x00000000 到 0xBFFFFFFF（3GB），而操作系统内核占用高位的 1GB 地址空间（0xC0000000

到 0xFFFFFFFF）。

- **进程加载位置**: 用户进程通常加载到较低地址处，比如 0x08048000。
- **影响**: 由于地址中没有严格限制不能包含 0x00 字节，因此编写 Shellcode 和返回地址时的限制较少。

Windows 系统

- **描述**: 在 32 位 Windows 系统中，虚拟地址空间被划分为用户模式和内核模式。用户模式占用 0x00000000 到 0x7FFFFFFF（2GB），内核模式占用 0x80000000 到 0xFFFFFFFF（2GB）。
- **用户进程加载位置**: 用户进程的可执行文件通常加载到 0x00400000 地址处。
- **影响**: 由于用户进程地址空间的很多位置都包含 0x00 字节（特别是低位地址），编写 Shellcode 和返回地址时会遇到更多挑战，因为很多情况下输入处理函数会认为 0x00 是字符串的结束符，导致截断攻击载荷。

返回地址中的 0x00 字节问题

在 Windows 系统中，由于用户进程的加载位置和地址空间的特点，返回地址中可能会包含 0x00 字节。举个例子，如果返回地址是 0x00401234，那么其中包含的 0x00 字节可能会导致攻击失败，因为处理输入的函数可能会把这个字节当作字符串结束符。这就需要在攻击载荷构建中采取特殊处理方法，以避免或绕过 0x00 字节。

Shellcode 模式解释

模式 1: NNNNSSSSAAAA

- **NNNN**: NOP 滑道（No Operation）。这是由许多 NOP 指令（如 0x90）组成的区域，使得程序执行到这里时会跳过这些指令而不执行任何操作，继续向前执行，增加攻击的成功率。
- **SSSS**: Shellcode。这是实际的恶意代码，当执行流跳转到这里时，会执行攻击者预期的操作，如打开一个 shell。
- **AAAA**: 覆盖的返回地址。当缓冲区溢出时，程序的返回地址被覆盖为这个地址，导致执行流跳转到 NOP 滑道或 Shellcode 的位置。

模式 2: NNNNASSSS

- **NNNN**: NOP 滑道。
- **A**: 目标地址的高位部分，通常是为了避开 0x00 字节。
- **SSSS**: Shellcode。

详细解释

1. **NOP 滑道（NNNN）**：
   - **目的**: 增加成功率。通过填充大量 NOP 指令，使得即使返回地址不精确指向 Shellcode 的开头，只要指向 NOP 滑道的任何位置，执行流会顺利滑向 Shellcode。
   - **实现**: 典型的 NOP 指令在 x86 架构上是 0x90。
2. **Shellcode（SSSS）**：
   - **目的**: 实际的恶意代码。当执行流跳转到这里时，会执行特定任务，如打开一个 shell、修改文件或其他恶意操作。
   - **编写**: Shellcode 通常用汇编语言编写，并转换为机器码。为了避免包含 0x00 字节，攻击者需要特别设计和优化 Shellcode。
3. **覆盖的返回地址（AAAA 或 A）**：
   - **目的**: 攻击者需要将这个地址放置在缓冲区中，覆盖原本的返回地址，使得程序返回时跳转到 NOP 滑道或 Shellcode。
   - **挑战**: 在 Windows 系统中，返回地址可能包含 0x00 字节，需要特别设计以避免这种情况。可以通过寻找没有 0x00 字节的合适地址，或通过其他技巧来避开 0x00 字节。

示例

假设攻击者发现一个 Windows 程序的缓冲区溢出漏洞，并且程序的返回地址位于 0x00401234。攻击者可能会构建一个攻击载荷如下：

```
AAAA...AAAA（64 字节填充） + NOP 滑道 + Shellcode + 0x00401234
```

如果发现 0x00401234 地址中的 0x00 字节导致问题，攻击者可能需要选择另一个没有 0x00 字节的地址，或调整攻击载荷，使得 NOP 滑道和 Shellcode 位于更高的内存地址。

总结

通过理解不同操作系统的内存布局及其对栈溢出攻击的影响，攻击者可以更有效地设计和实施攻击。在 Windows 系统中，避免 0x00 字节是一个关键挑战。利用 NOP 滑道和精心编写的 Shellcode，可以提高攻击成功率。