|  |  |  |  |
| --- | --- | --- | --- |
| **《计算机网络》实验报告** | | | |
| **实验编号** | 实验8 | **实验名称** | DNS |
| **姓名** | 李彦浩 | **学号** | 202100300063 |
| **班级** | 工业软件班 | **成绩** | （空着） |
| 1. **实验目的**   DNS (Domain Name System)是将域名转换为IP地址的系统和协议。本次实验将探究DNS协议。 | | | |
| 1. **实验要求**   wireshark：使用wireshark软件工具用于捕获和检查数据包跟踪。。  Browser：寻找并抓取网页资源。  Dig:Dig是一个灵活的命令行,用于查询远程DNS服务器的工具。 | | | |
| 1. **实验内容**   例子：   1. Network Setup 2. Manual Name Resolution 3. Capture a Trace 4. Inspect the Trace 5. Details of DNS Message 6. DNS Response Time 7. Explore Your Network | | | |
| 1. **实验过程**  Network Setup(搭建网络) 在一个典型网络中，你的计算机和一个本地的DNS命名服务器交互来将域名解析为IP地址。这台本地的DNS服务器可能是你所在内网的另一台主机。它又和远程的DNS服务器交换报文来完成解析。如下图：  0  本次实验遵循一条假设：我们在计算机上收集到的链路踪迹只能看到和本地DNS的交换报文，而非远程的。 Step 1:Manual Name Resolution(手动域名解析) 在搞明白我们的计算机是如何使用DNS之前，先看一下本地的DNS是如何解析域名的，其实就是如何和远程DNS服务器交互的。现在我们假装自己就是本地DNS，然后通过dig这个工具想远程DNS发送请求。  挑一个需要解析的域名，比如我挑了  www.baidu.com  怎么解析呢？就是通过搜索网络来获取根DNS服务器的IP地址。  例如，维基百科关于根域名服务器的文章包含了从a到m的根域名服务器的IP地址。其中任何一个都可以，因为它们持有相同的信息。需要这些信息才能开始名称解析过程，并在配置名称服务器时提供这些信息。  通过dig来向一个根域名服务器发起请求，完成解析的第一步。假定我们没有缓存的信息以便从根服务器下的某一级开始解析。  dig命令的格式形如：  dig @aa.bb.cc.dd domainname  这将向一个域名服务器发送一条给定域名的请求。[如果你想通过IP地址为198.41.04的服务器来查询www.uwa.edu.au](http://xn--IP198-fq1hp2ehz1aeep4v0xxsisql3jg7a.41.xn--04www-m86ht9t5l1bywa02biy0huu5c.uwa.edu.au)，dig命令应该写成：  dig @198.41.04 www.uwa.edu.au  从根域名服务器响应的内容不会提供这个域名对应的完整IP。而是告诉我们应该联系哪个子DNS。  比如，根DNS挑了一堆知道.au域名对应的IP地址的子DNS服务器，然后把它们的IP地址响应给我们。  从这一堆子DNS里，我们先挑按字母序最靠前的，这可能选出来一堆子DNS，然后我们再挑IP地址按数字序最靠前的，然后递归地使用dig，直到拿到完整的IP地址。  画出一幅以上过程的流程图。  1 Step 2:Capture a Trace 按照以下步骤捕获DNS流量：   1. 关闭所有浏览器窗口。浏览网页将解析域名以便连接到远程服务器，这将产生DNS流量。 2. 启动wireshark。过滤器：   udp port 53  DNS流量通常采用UDP协议传输，端口号53。  仿照之前的步骤重复dig指令。将看到DNS请求响应的数据包。可能会有一些本地计算机产生的背景DNS流量，但影响不大，这些流量占比比较少。  这里我选择了baidu.com这个域名，由于不知道百度的DNS根节点IP，所以上网搜了一下：  2  这个是百度官方认证的DNS服务IP：  180.76.76.76  但这个IP直接给出了域名解析的结果，也就是百度服务器的IP。当然了，递归一次也叫递归。  3  等待一小段时间，然后打开浏览器，浏览多个之前没访问过的网页，这将解析域名，产生DNS流量。 Step 3:Inspect the Trace DNS帧头在IP和UDP帧头之后。这是因为DNS报文携带在UDP之内，而UDP又是IP包中的段。  点开DNS帧头，有以下字段：  Transaction ID:用来连接请求和对应的响应的ID，一个DNS请求和一个DNS响应的Transaction ID是一致的。  4  Flags:表明DNS报文是查询还是响应，以及一些其他的细节。  5  Number of query/answer/authority/additional records:字面意思，DNS帧头到此结束。  6  DNS帧头之后，是报文的剩余部分，包括了查询次数、回答、权限和额外记录。通常仅有一次查询，即我们想要解析的域名的IP地址。  但同时常常有许多其他记录，按类别区分。比如Authority块里面都是authority records。  每种查询都有一个类型码(Type code)标识查找记录的类型是否是一个IP地址或者是其他什么东西。其他记录也有一个状态码表明它携带的是主机的IP地址还是域名服务器的名称还是其他别的什么东西。  单条记录的格式与它的类型相关。DNS报文被设计成能和UDP报文兼容的形式。  7  wireshark可能展示一些其他信息。比如响应中携带的数据包的数量或者是DNS报文交换的响应时间。但实际上DNS报文中没有任何这些信息。 Step 4:Details of DNS Messages(DNS报文细节) 查看dig指令对应的第一条DNS查询报文，展开它的DNS帧头。一般来说这个请求报文是捕获的数据包中的第一个，后面还有一些dig指令产生的DNS数据包，再后面就是浏览器产生的DNS流量。无视掉那些背景DNS流量。 Q1:Transaction ID有多少个bit?基于这个长度，猜测是否可能并发的流量会使用同样的Transaction ID。 8  4B = 32bit。一般来说同一组query-response的DNS报文使用同一个Transaction ID。  9  10 Q2:Flags字段中的哪一位标识了DNS报文是查询还是响应？这个字段的值是什么？ 第一位。0对应query，1对应Response。  11  12 Q3:整个DNS帧头有多少个字节？ 我捕获到的帧总共59B，但负载有15+32=47B，也就是说帧头总共有12B。  13  14  现在查看dig指令下的DNS查询对应的响应数据包。  一开始的响应应该提供了另一个更进一步的域名服务器，但并不是最终的答案。  而每个响应数据包中都在Query块中包含了最初的查询。同时也会包含下一个要连接的域名服务器的名称（可能有一堆这样的服务器），随后是这些服务器的IP地址。  在最后的响应中会包含域名对应的IP地址，这就是查询的最终结果。  查看DNS响应报文体，回答： Q4:对于初始响应来说，在哪个块/section中包括了域名服务器的名称？携带域名服务器名称的记录类型是什么？ 在Additional Records这个块里，我捕获到的帧中，这个名字就叫“<Root>”。类型是OPT（可选的意思，Optional)。  15 Q5:相似的，域名服务器对应的IP地址在哪个块中，记录类型是什么？ Answers中。由于我的查询中没有中间的DNS服务器节点，所以直接把目标IP查出来了。  16 Q6:对于最终的响应来说，域名对应的IP地址在哪个块里？ Answers中。  16 **Step 5:DNS Response Time(DNS响应时间)** 这回查看浏览器发出的DNS查询的DNS响应时间。与dig指令相对的，这种DNS流量比较常见，也就是说它发了单个查询并直接在响应中收到了解答。响应时间就是从本地DNS发出查询开始，到从本地DNS收到响应的延迟时间。  按照以下步骤生成一个DNS响应时间的IO图。IO图是wireshark通过Statistics模块提供的标准图表。  对于X轴，调整时钟间隔以及每个时钟查看的像素点。一般来说1s，别的足够小的值也ok。  对于y轴，把单位替换为“Advanced”，默认的是Packet/Tick。”Advanced”是一个特殊的关键词，允许我们查看图上不同的数据。一旦选中，“Calc:”属性将会展示选中的数据。  在calculation box中输入dns.time。并将下拉菜单设置为”MAX(\*)”。  dns.time是一个wireshark根据查询和响应报文计算出的字段。MAX(\*)意思就是选取这个计算属性中的最大值。查看最大值我们可以发现异常值。AVG(\*)展示的是平均值。  敲回车，然后点击“Graph”按钮。  可以看到许多小的DNS响应时间，这是因为大部分远程主机的IP地址已经缓存到了本地域名服务器中。以及一些分散的较大值，这是因为这些域名对应的IP还没被缓存。  我现在的高版本Wireshark已经没有Advanced这个选项了，但幸运的是它仍然提供了计算属性，如下图：  17 Explore Your NetworkQ1:查看其他类型的DNS记录。例如MX对应邮件服务器的域名，AAAA对应IPv6地址的域名。 抓到了类型为HTTPS，值为65的DNS帧。  18 Q2:谷歌提供了”Google Public DNS”。查看这个服务器，然后根据配置说明测试一下。看看它是否比已有的DNS服务快。 由于我无法翻墙（在中国这是违法的），因此该小问跳过。 Q3:DNS反查是根据IP地址查域名。一般用于安全检查。尝试一下DNS反查服务。 我使用一个在线工具  <https://www.infobyip.com/>  对IP 8.8.8.8进行反查。  19  发现它是google的dns服务器。 Q4:DNSSEC是DNS的安全扩展服务。它使用了附加的DNS记录类型并返回密钥以及签名信息，域名服务器可以检查响应的权限。使用dig命令测试一下DNSSEC，需要在命令行中添加”+dnssec”来打开这项服务。 命令：  dig +dnssec baidu.com  20  可以看到比一般的dig多出了一些OPT可选项。 | | | |
| **四．实验心得**  本次实验我根据实验指导书的指引，探究了DNS协议的细节。首先我接触到了一个域名解析工具dig，这个工具现在在win上没有，于是我来到linux虚拟机下进行实验。实验指导书让我先假想自己就是本地DNS服务器，然后采用递归dig的方式来进行手动域名解析。配合wireshark抓包，我抓到了自己递归dig命令时产生的DNS，随后逐一分析了帧头中协议的细节。  之后我采用Wireshark提供的流量解析功能，列出了DNS的IO流量图，这有助于我更直观地查看DNS流量，之后我在上面加入了一些计算统计属性，观察了DNS流量高峰期的状态。  最后我尝试探索了不同类型的DNS帧，比如值为65的HTTPS类型以及字段值为5的alias类型。之后我在网上找到了一些DNS反查的工具，通过提供IP地址来反向解析域名。并且尝试了dnssec服务，观察了它与一般dig命令的不同。  这次实验让我对DNS协议有了更深的理解，为将来计算机网络的学习打下了扎实的基础。 | | | |