第 5 章

亭台楼阁——应用层协议介绍

通常网络故障的排错指令集中在网络层，而针对应用层的检测手段较少，遇到应用故障，还需要研发人员配合才能进行应用服务器的检查。网络工程师在熟练掌握网络流量分析后，可提高对应用层的分析能力。

5.1 超文本的传输方式——HTTP

HTTP(Hyper Text Transfer Protocol，超文本传输协议)是用于从万维网(World Wide Web，WWW)服务器传输超文本到本地浏览器的传送协议。

1960 年，美国人 Ted Nelson 构思了一种通过计算机处理文本信息的方法，并称之为超文本(Hyper Text)，这成为 HTTP 标准架构的发展根基。Ted Nelson 组织协调万维网协会(World Wide Web Consortium，W3C)和互联网工程工作小组(Internet Engineering Task Force，IETF)共同合作研究，最终发布了一系列的 RFC，其中著名的 RFC2616 定义了 HTTP 1.1，HTTP 新版本为 2.0，由于目前 HTTP 1.1 应用还较为广泛，因此本节主要介绍 HTTP 1.1。

5.1.1 HTTP 简介

HTTP 是一个应用层协议，基于 TCP/IP 来传递数据(HTML 文件、图片文件、查询结果等)。HTTP 在 OSI 模型中的位置如图 5.1 所示。

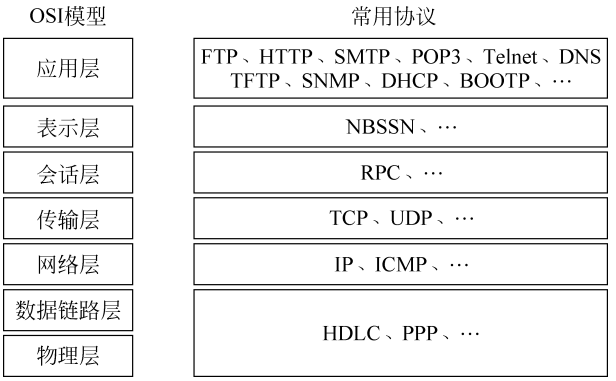


图 5.1 HTTP 在 OSI 模型中的位置

HTTP 使用 C/S(客户端/服务器)架构。客户端使用 Web 浏览器向 Web 服务器发送 HTTP 请求。Web 服务器根据接收到的请求向客户端发送对应的响应信息。HTTP 的交互过程如图 5.2 所示。

图 5.2　HTTP 的交互过程

由于现实网络环境较为复杂,在客户端和服务器中间可能存在多个中间层,比如代理、网关、隧道等。TCP 可以为 HTTP 避免上述中间层带来的网络影响。所以通常情况下,HTTP 使用 TCP 进行可靠传输。

【经验分享】 HTTP 不仅可以基于 TCP 进行数据传递,还能在任何其他互联网协议上,甚至在其他网络上实现其功能。

在常见的网络环境下,要进行 HTTP 通信,必须先建立 TCP 会话,然后在 TCP 会话中传递 HTTP 数据,RFC2616 文档规定了 HTTP 的默认端口号为 80,因此在建立 TCP 会话时,客户端采用随机非特定协议的端口主动向服务器的 80 号端口发起连接。

当访问 baidu.com 这类网址时,用户输入的是便于人们记忆的域名,并不是真正的网站服务器 IP 地址,所以在建立 TCP 会话之前,首先进行的工作是通过 DNS 协议的解析工作将域名变成 IP 地址。

综上所述,一次完整的 HTTP 客户端访问服务器的过程如图 5.3 所示。

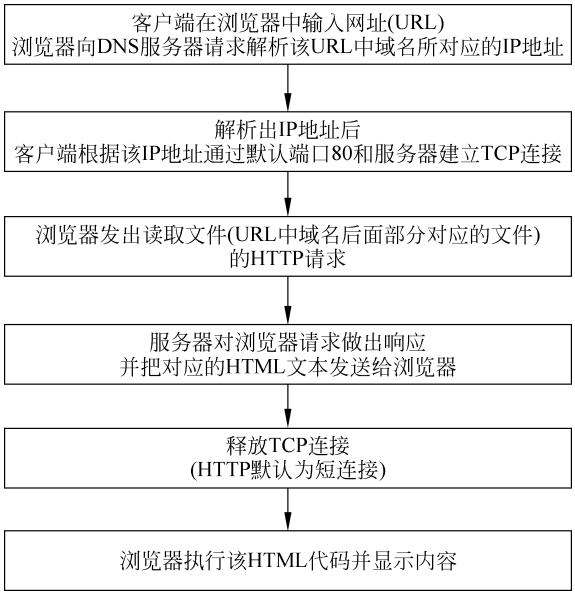


图 5.3　HTTP 访问服务器全过程示意图

5.1.2 HTTP URI

如何访问网络上某一指定的 HTTP 资源？例如网页、图片、音乐、视频等资源。HTTP 使用统一资源标识符(Uniform Resource Identifier,URI)来对资源进行标识,并建立连接和传输数据。

URI 是资源表示方式的一种统称,细分下来有 URL 和 URN 两种资源表示方式。统一资源定位符(Uniform Resource Locator,URL)是互联网上用来标识某一处资源的地址,统一资源名称(Uniform Resource Name,URN)是互联网上用来标识某一处资源的名称。

URL 的格式如图 5.4 所示。

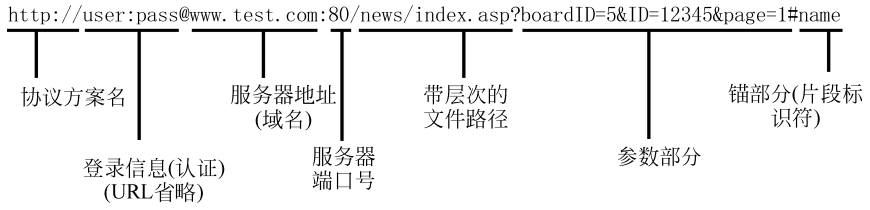


图 5.4 URL 的格式

各字段的作用说明如下：

(1) 协议部分。该 URL 的协议部分为“http:”,这代表访问网页使用的协议是 HTTP。在互联网中还可以使用多种其他协议,如 FTP、MAIL、Telnet 等。该部分以“//”作为结束标记。

(2) 认证部分。认证不是 URL 必须使用的部分,在使用中可以省略,该 URL 的认证部分为 user:pass,冒号前为用户名,冒号后为密码,当客户访问一些需要认证才能访问的页面时,可以直接在 URL 中填入用户名和密码,例如,启用了 HTTP 基本认证的页面、启用了认证的 FTP 页面等。如果访问的页面无须认证,则在 URL 中不需要此部分,如果使用认证部分,则“@”符号作为本部分内容的结束标记。

(3) 域名部分。该 URL 的域名部分为 www.test.com。在一个 URL 中,也可以使用 IP 地址作为域名使用。

(4) 端口部分。端口不是一个 URL 必须使用的部分,在使用中可以省略,若在域名后省略了端口,将使用协议的默认端口进行访问,例中 HTTP 的默认端口为 80。如果指定对某端口进行访问,则需将端口输入在域名后面,域名和端口之间使用“:”作为分隔符。

(5) 虚拟目录部分。从域名后的第一个“/”开始到最后一个“/”为止,是虚拟目录部分。虚拟目录也不是一个 URL 必须使用的部分。若只访问 Web 网站的首页,则无须在 URL 中输入目录部分,本例中的虚拟目录是“/news/”。

(6) 文件名部分。从域名后的最后一个“/”开始到“?”为止,是文件名部分。如果没有“?”,则从域名后的最后一个“/”开始到“#”,是文件部分。如果没有“?”和“#”,那么从域名后的最后一个“/”开始到结束,都是文件名部分。本例中的文件名是“index.asp”。文件名部分也不是一个 URL 必须使用的部分,如果省略该部分,则使用默认的文件名。

(7) 参数部分。“?”和“#”之间的部分为参数部分，又称搜索部分、查询部分。本例中的参数部分为“boardID=5&ID=24618&page=1”。URL 中允许有多个参数，参数与参数之间用“&”作为分隔符。

(8) 锚部分。从“#”开始到最后，都是锚部分(片段标识符)。在一个页面中，可以存在多个锚，访问这些锚将让网页滚动到特定位置，类似于 Word 文档中的目录功能，单击目录即可跳转到对应的章节。本例中的锚部分是 name。锚部分也不是一个 URL 必须使用的部分。

5.1.3　HTTP 请求包

在了解了 HTTP 如何使用 URI 标记网络上的资源以后，当访问某一网络资源时，需要向该资源发送 HTTP 请求。本节将详细介绍 HTTP 请求各字段的内容。

与传输层 TCP、网络层 IP、数据链路层 ARP 等协议一样，HTTP 也具有自己发送请求的特定格式，但区别在于 TCP/IP/ARP 格式中的字段长度是固定的，而 HTTP 格式的字段长度不是固定的，同时因为 HTTP 采用 ASCII 编码发送字符的方式传递数据，所以需要一些标记用以区分哪些字符属于哪个字段(比如空格、空行)。

HTTP 请求包由客户端发送给服务器，请求消息包括请求行、请求报头、空行、请求数据 4 个部分。图 5.5 展示了经过 ASCII 解码的 HTTP 请求报头，使用记事本工具能够更加直观地观察到 HTTP 请求的内容。

图 5.5　HTTP 请求包格式

通过图 5.5 可以看出 HTTP 四个部分的区别。接下来详细介绍每部分的作用。

1. 第一部分：请求行

请求行包含三个部分：请求类型、请求路径和 HTTP 版本。POST 说明请求类型为 POST，“/”为要访问的资源路径，最后一部分说明使用的是 HTTP 1.1 版本。请求行以十六进制 0D 0A 标记换行，表示该行结束。

2. 第二部分：请求报头

第二行以及后续内容用于声明请求中附加的参数。从第二行起为请求报头，请求报头可能会有多行，但每一行的格式都应为“参数名称：参数内容”。例如：

- Host 参数将指出客户端请求的网站名称。
- User-Agent 参数将指出客户端请求时使用的浏览器版本。
- Content-Type 参数将指出客户端请求的数据类型。
- Content-Length 参数将指出客户端请求第四部分的长度。

【注意】 不同浏览器在请求时会使用不同的请求报头字段，某些浏览器的请求报头

支持自定义,详细的报头字段将在5.1.5节进行介绍。

3. 第三部分:空行

请求报头后面的空行是必需的。空行用于区分第二部分与第四部分,即使第四部分的请求数据为空,也必须有空行。

4. 第四部分:请求数据

请求数据也叫实体、主体、body,可以传递任意的其他数据,如网页、图片、音乐等。

5.1.4 HTTP请求方法

前文介绍过,HTTP请求包中首行的首个字段为HTTP请求方法,如图5.6所示,该请求使用GET请求方法。

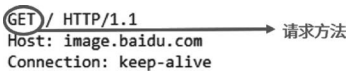


图5.6 HTTP GET请求方法

为了实现不同的访问需求,如上传、下载、获取、修改等,HTTP共支持9种方法进行请求,其中HTTP 1.0定义了三种请求方法:GET、POST和HEAD,其余6种请求方法为HTTP 1.1所定义。本节详细介绍这些HTTP的请求方法。HTTP的请求方法及其功能如表5.1所示。

表5.1 HTTP请求方法汇总

方 法	功 能
GET	请求获取Request-URI所标识的资源
HEAD	请求获取由Request-URI所标识的资源的响应消息报头
POST	在Request-URI所标识的资源后附加新的数据
OPTIONS	查询Web服务器的性能(探测服务器支持的方法)
DELETE	请求服务器删除Request-URI所标识的资源
PUT	请求服务器存储一个资源,并用Request-URI作为其标识
PATCH	是对PUT方法的补充,用来对已知资源进行局部更新
TRACE	跟踪到服务器的路径,主要用于测试或诊断
CONNECT	对通道提供支持

一个URI用于描述一个网络上的资源,常常被提及的HTTP中的PUT、DELETE、POST、GET方法就分别对应着对这个资源的增、删、改、查4个操作。

1. GET请求

GET请求类似于“查”,当用户通过浏览器访问某个网站时,发起的请求均是GET请求。

2. HEAD请求

与GET请求几乎是一样的,用户可以通过HEAD请求访问某个网站,此时服务器将只响应HTTP响应报头,不返回HTTP正文。该请求常用于测试超链接的有效性、是否可以访问以及最近是否更新,是一种探测性的访问。

3. POST请求

POST请求类似于“改”,当用户在某个网页内进行登录操作时,此时用户输入的用户名和密码将会通过POST请求发送到服务器,或当用户在某个网页内上传头像时,此时

头像文件也会使用 POST 请求发送。

4. OPTIONS 请求

通过 OPTIONS 对某个 URI 进行请求时，网站会返回当前 URI 所支持的 HTTP 请求方法，如 GET、HEAD、POST、DELETE 等，通常来说攻击者可能利用这个请求来对网站进行探测，以便实施下一步的动作。

5. DELETE 请求

类似于“删”，使用 DELETE 请求访问某个 URI 时，服务器会直接删除该 URI 下的文件。这是一种极其危险的方法，网站管理员往往会禁止对 DELETE 请求进行响应。

6. PUT 请求

类似于“增”，使用 PUT 请求的目的是为了通过浏览器向网站服务器上传文件，这也是一种极其危险的方法，网站管理员通常会禁止对 PUT 请求进行响应。

7. PATCH 请求

PATCH 请求和 PUT 请求类似，是 PUT 请求方法的补充，当使用 PATCH 进行请求时，目的是为了对服务器已经存在的资源进行部分内容的更新，与 PUT 一样，这是一种危险的 HTTP 请求方法，网站管理员通常会禁止对 PATCH 请求进行响应。

8. TRACE 请求

TRACE 请求用于测试，使用该请求访问某个网站时，该网站服务器会将收到的 TRACE 请求原样返回给客户端。

9. CONNECT 请求

CONNECT 请求多用于使用 HTTP 代理的情况，客户端向 HTTP 代理服务器发起一个 CONNECT 请求，HTTP 代理服务器向网页服务器建立一个连接。

5.1.5　HTTP 报头字段

在客户端与服务器之间进行通信的过程中，无论是 HTTP 请求包还是 HTTP 响应包都会携带多个报头字段，HTTP 报头字段是由报头字段名和字段值构成的，中间用冒号“:”分开，如 Content-Length:40。这些报头字段能起到传递额外重要信息的作用。RFC2616 一共定义了 47 种报头字段，随着使用需求的增加，HTTP 会更新、增加功能，因此还包括 Cookie、Set-Cookie 和 Content-Disposition 等在 RFC 中定义的其他报头字段，它们的使用频率也很高。在 RFC4229 中，归纳的正式和非正式的报头字段共计 133 种。

某些字段用于传递 HTTP 请求的额外信息，某些字段用于传递响应的额外信息，某些字段用于传递实体的额外信息，还有一些字段则是通用的，因此 HTTP 的报头字段被归为如下 4 类：

- 请求报头字段。
- 响应报头字段。
- 实体报头字段。
- 通用报头字段。

报头字段属于 HTTP 包格式中的“第二部分”，在 HTTP 包中所处的位置如图 5.7 所示。

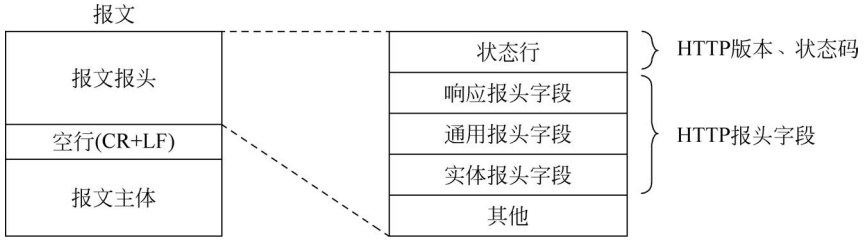


图 5.7 HTTP 报头字段位置示意图

RFC2616 中规定了 47 种报头字段,如表 5.2 所示。

表 5.2 HTTP 报头字段

通用报头字段	说 明
Cache-Control	控制缓存的行为
Connection	逐跳报头、连接的管理
Date	创建包的日期时间
Pragma	包指令
Trailer	包末端的报头一览
Transfer-Encoding	指定包主体的传输编码方式
Upgrade	升级为其他协议
Via	代理服务器的相关信息
Warning	错误通知
请求报头字段	**说 明**
Accept	用户代理可处理的媒体类型
Accept-Encoding	优先的内容编码
Accept-Charset	优先的字符集
Accept-Language	优先的语言
Authorization	Web 认证信息
Expect	期待服务器的特定行为
From	用户的电子邮件地址
Host	请求资源所在的服务器
If-Match	比较实体标记(ETag)
If-None-Match	排除实体标记(ETag)
If-Range	请求部分实体标记
If-Modified-Since	比较资源的更新时间
If-Unmodified-Since	与 If-Modified-Since 相反
Max-Forwards	最大传输逐跳数
Proxy-Authorization	代理服务器要求客户端的认证信息
Range	实体的字节范围请求
Referer	对请求中的 URI 的原始获取
TE	传输编码的优先级
User-Agent	HTTP 客户端程序的信息

续表

响应报头字段	说　明
Accept-Ranges	是否接受字节范围请求
Age	推算资源创建经过时间
Etag	资源的匹配信息
Location	令客户端重定向至指定 URL
Proxy-Authenticate	代理服务器对客户端的认证信息
Retry-After	对再次发起请求的时机要求
Server	HTTP 服务器的安装信息
Vary	代理服务器缓存的管理信息
WWW-Authenticate	服务器对客户端的认证信息
实体报头字段	**说　明**
Allow	资源可支持的 HTTP 方法
Content-Encoding	实体主体适用的编码方法
Content-Language	实体主体的自然语言
Content-Length	实体主体的大小(字节)
Content-Location	替代对应资源的 URI
Content-MD5	实体主体的包摘要
Content-Range	实体主体的位置范围
Content-Type	实体主体的媒体类型
Expires	实体主体过期的日期时间
Last-Modified	资源的最后修改日期时间

还有 RFC2616 没有列出的报头字段，如 Cookie、Set-Cookie 和 Content-Disposition 等。接下来详细介绍 HTTP 中常用的一些字段的作用，更加详细的 HTTP 字段的介绍请参见 RFC4229。

Content-Length：描述了 HTTP 实体(HTTP 包中的第四部分)的长度，单位为字节。

Content-Type：描述了 HTTP 实体传输的数据类型，如图片、音频、视频、文字、程序等。

Connection：描述了 HTTP 交互之后如何处理 TCP 连接，HTTP 在设计之初是一个“无连接”(又称“短连接”)的协议，“无连接”的含义是限制每个 TCP 连接只处理一个 HTTP 请求。服务器处理完客户的请求并收到客户的应答后，即断开连接。在设计之初，采用这种方式可以节省传输时间。但时至今日，Web 页面的规模越来越庞大，HTTP 需要一种方法来实现长连接的 HTTP 通信。因此，通过 Connection 来声明 HTTP 的连接是否需要保持，以实现长连接的 HTTP 通信。客户端和服务器声明 Connection：keep-alive 表示希望保持连接，当声明 Connection：closed 表示希望断开连接。

Cookie 和 Set-Cookie：HTTP 是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着在某购物网站登录后，刷新页面，由于协议的无状态特性，将导致刚输入的登录信息不被服务器所记忆，从而造成永远无法登录成功的死循环。在协议设计之初，因为当时的 Web 页面规模小、架构简单，所以采用无状态方式进行工作。目前

可以通过 Cookie 的方式来弥补 HTTP 无状态的缺点。在客户端访问服务器时,服务器通过 Set-Cookie 字段为客户端分配一个无意义的字符串,客户端下次访问时,在 HTTP 请求包中通过 Cookie 字段携带这个字符串,服务器通过这个字符串辨别请求来自哪一个客户端,使 HTTP 从无状态协议变为有状态协议。

5.1.6 HTTP 响应包

HTTP 的报头字段不止出现在 HTTP 请求包中,也会出现在 HTTP 响应包中。一般情况下,服务器接收并处理客户端发过来的请求后会返回一个 HTTP 的响应消息。HTTP 响应也由 4 个部分组成,分别是状态行、响应报头、空行和响应正文。

图 5.8 展示了经过 ASCII 解码的 HTTP 请求头,使用记事本工具能够更加直观地观察到 HTTP 请求的内容。

图 5.8 HTTP 响应包格式

通过图 5.8 可以明显地看出 HTTP 响应包也可以分为 4 个部分,与 HTTP 请求包的区别不大。接下来将详细介绍这 4 个部分的区别。

1. 第一部分:状态行

由 HTTP 版本号、状态码、状态消息三部分组成。HTTP/1.1 表示 HTTP 版本为 1.1,200 表示状态码,OK 表示状态消息。

2. 第二部分:响应报头

与请求部分的请求报头类似,响应报头用来说明客户端要使用的一些附加信息。如图 5.8 所示,Date 生成了响应的日期和时间,Content-Type 指定了 MIME 类型的 HTML (text/html),编码类型是 UTF-8。

3. 第三部分:空行

消息报头后面的空行是必需的,用于区分 HTTP 响应中的第二部分和第四部分。

4. 第四部分:响应正文

服务器返回给客户端的文本信息。空行后面的 HTML 部分为响应正文。

5.1.7 HTTP 状态码

一旦收到 HTTP 请求,服务器会向客户端返回 HTTP 响应包,响应包的状态行包括 HTTP 版本、状态码、状态消息,比如“HTTP/1.1 200 OK”和响应的消息。响应消息的消息体可能是请求的文件、错误消息或者其他信息,如图 5.9 所示。

HTTP 的通信有成功、失败、拒绝、其他等几种情况,服务器会在响应消息中发送一

图 5.9　HTTP 响应中的状态码和状态消息

个“状态码”,用不同的状态码来表示不同的含义。如图 5.9 中的“200”就是一个最常见的状态码,表示请求成功。状态码后面的“OK”为状态消息,配合状态码使用。

状态码由三位数字组成,共分为 5 种类别,如表 5.3 所示。状态码的第一个数字定义了响应的类别。

表 5.3　HTTP 状态码分类

状　态　码	类　　别	类 别 短 语
1 **	指示信息状态码(Informational)	接收的请求正在处理
2 **	成功状态码(Success)	请求正常处理完毕
3 **	重定向状态码(Redirection)	需要进行附加操作以完成请求
4 **	客户端错误状态码(Client Error)	客户端的请求有误或请求无法实现
5 **	服务器错误状态码(Server Error)	服务器处理请求出错

常见的 HTTP 状态码如表 5.4 所示。更多的 HTTP 状态码请参考 RFC2616、RFC2518、RFC2817、RFC2295、RFC2774、RFC4918。

表 5.4　常见的 HTTP 状态码

状态码	原 因 短 语	含　　义
200	OK	客户端请求成功
204	No Content	已处理成功,但没有实体主体内容返回
206	Partial Content	客户进行了范围请求(如请求资源的指定部分),服务器执行了这部分请求
301	Moved Permanently	资源(网页等)被永久转移到其他 URL
302	Found	临时重定向(请求的资源使用了新的 URI,希望本次用户用新的 URI 进行访问)
304	Not Modified	请求的资源允许访问,且符合条件,可直接从本地缓存读取(304 响应不包含响应正文)
400	Bad Request	客户端请求有语法错误,不能被服务器所理解
401	Unauthorized	请求未经授权,这个状态代码必须和 WWW-Authenticate 包头域一起使用
403	Forbidden	服务器收到请求,但是拒绝提供服务
404	Not Found	请求资源不存在,如输入了错误的 URL
500	Internal Server Error	服务器发生不可预期的错误
503	Sever Unavailable	服务器当前不能处理客户端的请求,一段时间后可能恢复正常

5.1.8 HTTP 分析方法

科来 CSNAS 提供三种分析 HTTP 数据包的方式,分别是数据包解码、数据流解码、HTTP 日志方式,可以按需选择分析 HTTP 数据包的方法,也可以按需将多种分析方法组合使用。

1. 数据包解码

在“数据包”视图直接观察 HTTP 请求、应答包的解码,该方式比较适合初学者。在解码界面中,科来 CSNAS 会解释 HTTP 请求、应答包中的各个字段,如图 5.10 所示。

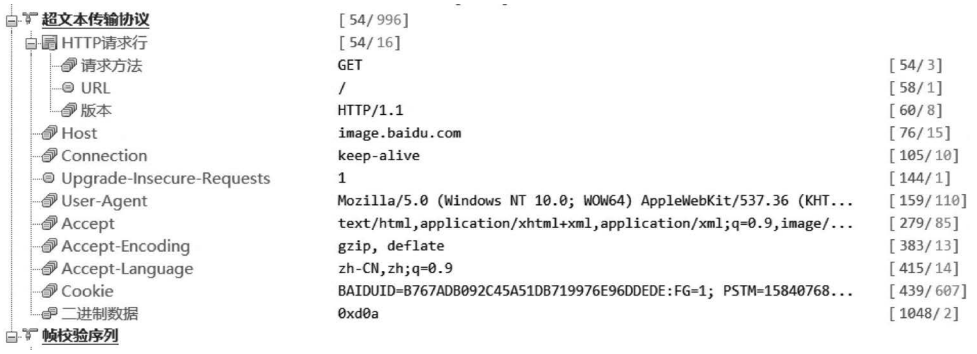


图 5.10 HTTP 包解码信息

通常网络中不可能只有 HTTP 数据包,在使用数据包解码对 HTTP 数据包进行分析时,会遇到其他干扰数据包太多的问题,比如分析 HTTP 流量时一定会同时出现 DNS、TCP 等其他协议的数据包。

如图 5.11 所示,科来 CSNAS 一共捕获到了 61394 个数据包,从截图中可以明显看出这其中包括一些 NBNS、DNS 协议的数据包,这些数据包对于 HTTP 来说均是杂音,影响分析效率。

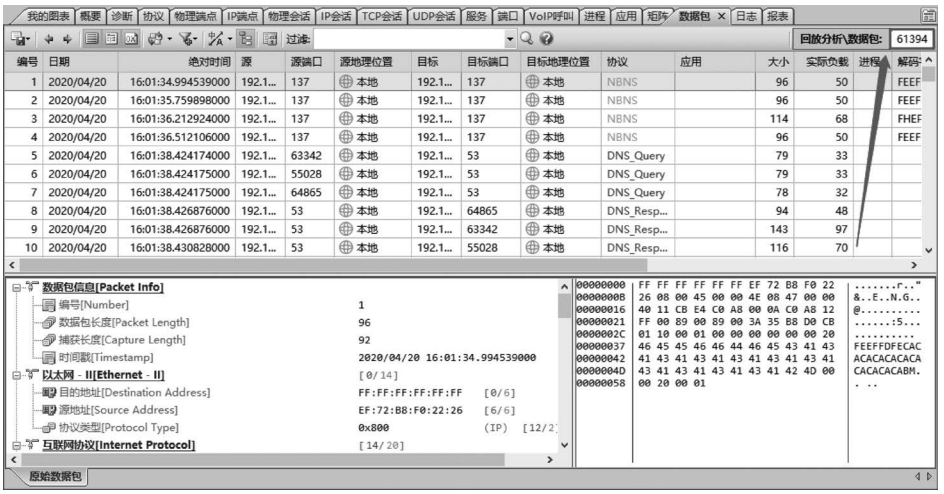


图 5.11 捕获到的数据包数量

此时可以对数据包进行过滤，有如下两种过滤方式：

(1) 使用端口进行过滤，如 80、21、25 等。需在过滤窗口中输入过滤语句：port＝80，如图 5.12 所示。

图 5.12　端口过滤语句

此时，软件显示的数据包数量从之前的 61394 变成了 60972，显示了所有源、目的端口为 80 的数据包。如需过滤源端口，语句为 srcport＝80，过滤目的端口的语句为 dstport＝80。

(2) 使用协议进行过滤，如 HTTP、FTP、SMTP 等。需在过滤窗口中输入过滤语句：protocol＝http，如图 5.13 所示。

图 5.13　协议过滤语句

此时，软件显示的数据包数量从之前的 61394 变成 38269。可以看到，使用两种不同的过滤方式得到的结果也不同，原因是第二种过滤方式排除了 TCP 的三次握手和四次断开数据包，在分析性能时会过滤掉连接不成功的会话。因此，在过滤时更推荐使用端口过滤的方式分析 HTTP 数据包。

在分析时需要注意，如果遇到需要一起分析三次握手、四次断开的情况，则不应该使用第二种方式，应尽量采用第一种端口过滤的方式。

2. 数据流解码

在“TCP 会话”视图观察 HTTP 的 TCP 会话，单击“数据流”直接观察 TCP 通信数据流，该方式能够更快速地分析这条 HTTP 的请求和响应，更适合对 HTTP 包格式熟悉的读者，信息如图 5.14 所示。

3. HTTP 日志

在“日志”视图选中左侧的“HTTP 日志”观察 HTTP 日志，这些日志全部是基于捕获到的数据包生成的，并以列的形式展示，每一列为一次 HTTP 请求/应答，用户可以基于 HTTP 日志功能快速浏览网络中出现的 HTTP 行为。当日志数量过多时，HTTP 日志支持以关键字作为条件进行搜索，快速列出某些特定的 HTTP 请求(如 POST 请求)，如图 5.15 所示。

当发现某条可疑的 HTTP 日志时，双击这条日志即可弹出新窗口，该窗口显示这条 HTTP 请求的 TCP 会话数据流，如图 5.16 所示。

学习了分析方法，还应该了解以下分析技巧：

(1) 在“协议”视图下可以快速查看 HTTP 的流量、网络连接、数据包等参数。

图 5.14 TCP 数据流解码

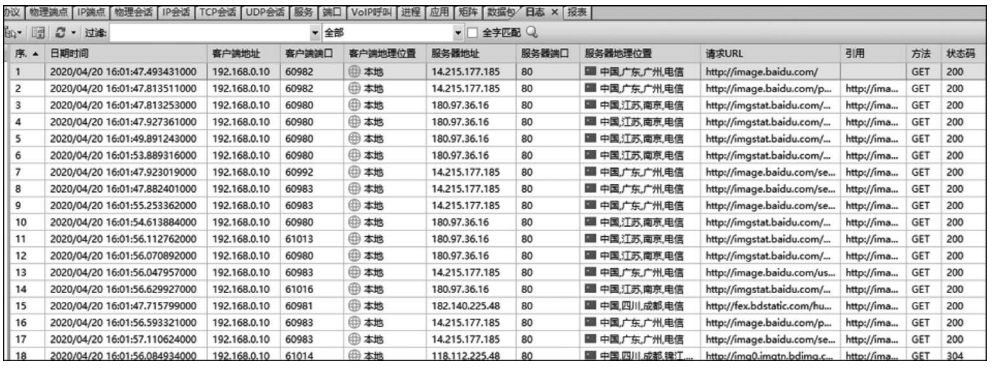


图 5.15 HTTP 日志

(2) 在“诊断”视图下可以查看是否存在关于 HTTP 的网络事件发生,如 HTTP 服务器响应慢、可疑的 HTTP 传输等。

(3) 在节点浏览器中,选择 HTTP,即可只显示 HTTP 相关的流量数据包,同时进入“概要”“诊断”视图单独针对 HTTP 进行分析。

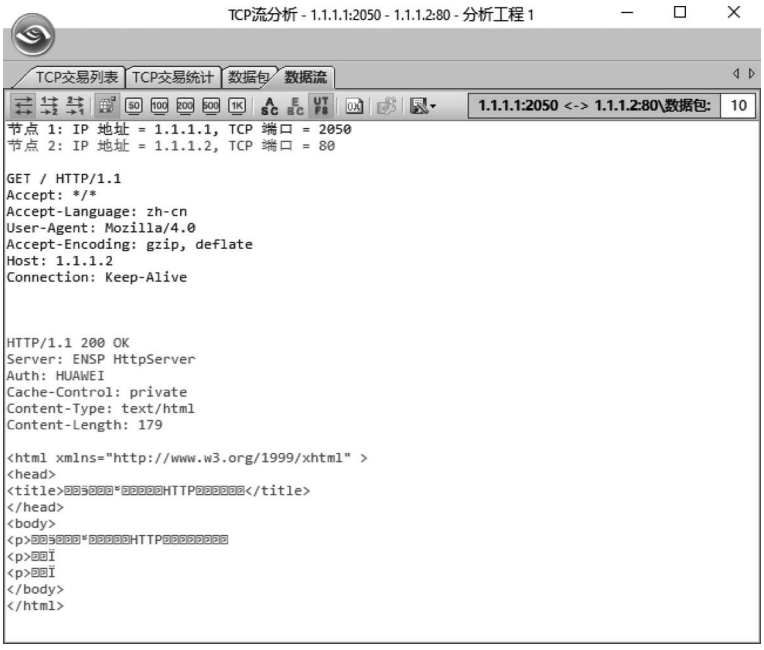


图 5.16　双击日志后弹出的 TCP 数据流分析窗口

(4) 通过第 4 章介绍的应用交易分析方法，分析 HTTP 的响应时间、传输时间、处理时间等参数，判断 HTTP 服务器的工作效率。

(5) 某些 HTML 页面在接收时需要同时打开多个 TCP 会话，为页面上的每一个对象和文本都建立一个 TCP 会话。

5.2　镖车与鸡粪——HTTPS

随着 HTTP 在互联网中的广泛运用，其设计时存在的以下缺陷也逐渐显露出来：

(1) 通信使用明文，内容可能会被窃听。

(2) 不验证通信方的身份，有可能遭遇伪装。

(3) 无法证明包的完整性，有可能信息已遭篡改。

HTTPS 应运而生，它是 HTTP＋加密＋认证＋完整性保护的综合体。

举例来说，古时由于治安条件不如今日好，镖局的做法是在镖车上涂满鸡粪，让人误以为这一镖车的货物没有实际价值，从而打消劫匪劫镖的企图。对于 HTTP 来说也是一样，如果能够在 HTTP 传输的内容上涂满“鸡粪”，让截获到 HTTP 通信内容的人无法发现内容中的奥秘，也就实现了防止窃听的目的；此外，需要一种方法来让 HTTP 能够验证通信方的身份，防止中间有人“偷梁换柱”，确保消息是从发送方那边发送过来的。这些都是在运输货物、信息时应该考虑的问题。

5.2.1　什么是 HTTPS

HTTPS 是以安全为目标的 HTTP 通道，并不是独立于 HTTP 的一个全新协议，而

是在 HTTP 的基础上添加了 SSL/TLS 握手以及数据加密传输,也属于应用层协议,如图 5.17 所示。

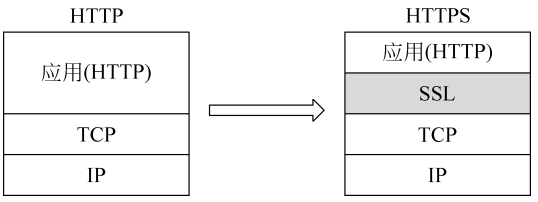


图 5.17 HTTP 与 HTTPS 对比

通常 HTTP 直接和 TCP 通信。当使用 SSL 时,则演变成先和 SSL 通信,再由 SSL 和 TCP 通信。所谓 HTTPS,其实就是身披 SSL 协议这层外壳的 HTTP。

5.2.2 HTTPS 建立连接的过程

整体来看,HTTPS=HTTP+SSL,在进行 HTTPS 交互时,应先建立 SSL 隧道,通过 SSL 隧道来对数据进行加密,然后在 SSL 隧道中传递 HTTP 数据。图 5.18 展示了 HTTPS 通信的过程,应先建立对应的 TCP 会话,然后在 TCP 会话中建立 SSL 加密通道,最终在加密通道中传递 HTTP 数据,使其加密,哪怕被外界截获,也不会造成信息失窃、冒用等损失。

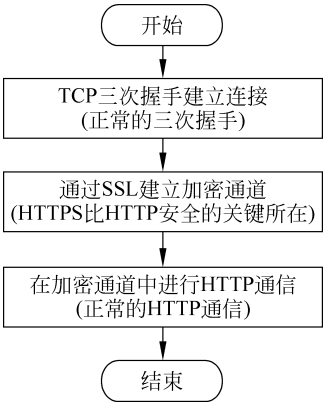


图 5.18 从数据包层面看 SSL 加密通道建立的过程

研究 HTTPS 的原理,关键是研究 SSL/TLS 协议。图 5.19 展示了从数据包层面看 SSL 加密通道建立的过程。

(1) 三次握手后,客户端通过发送 ClientHello 数据包开始 SSL 通信。在数据包中声明客户端支持的 SSL 的指定版本、加密组件列表(所使用的加密算法及密钥长度等信息的列表),以及一个客户端生成的随机数(Client Random)发送给服务器(用于后期组成通信密钥)。

(2) 服务器在收到 ClientHello 数据包后,会以 ServerHello 数据包作为应答。在数据包中声明使用的 SSL 版本(使用与客户端兼容的版本)、加密组件(从客户端发送的列表中筛选)以及一个客户端生成的随机数(Server Random)发送给服务器(用于后期组成通信密钥)。

(3) 服务器发送 Certificate 数据包。该数据包中携带服务器的公钥证书。

(4) 服务器发送 ServerHelloDone 数据包通知客户端,最初阶段的 SSL 握手协商部分结束。

(5) SSL 第一次握手结束之后,客户端以 ClientKeyExchange 数据包作为回应。该数据包中包含通信加密中使用的一种被称为 Pre-Master-Secret 的随机密码串(基于前面明文发送的随机数和客户端在本阶段再次产生的随机数综合计算得出)。该数据包已用

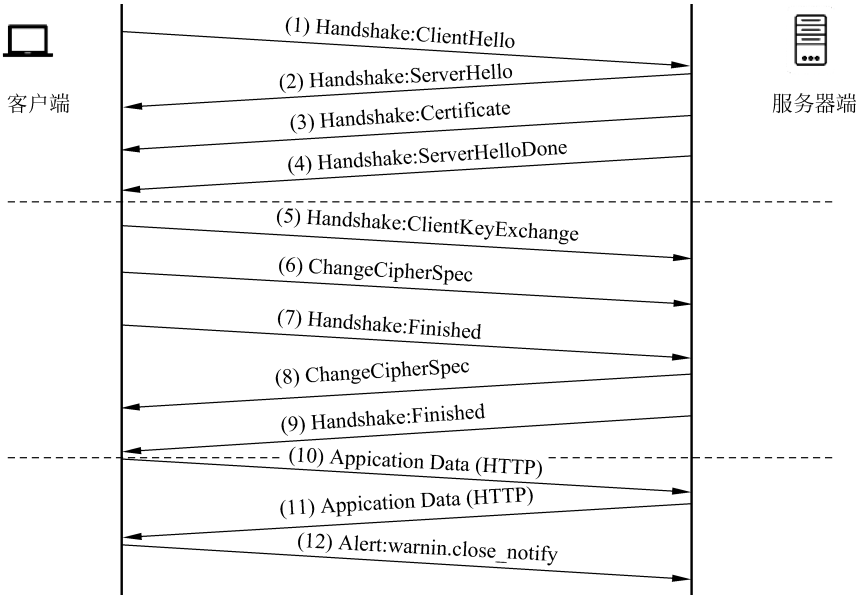


图 5.19　SSL 加密通道建立的过程

(3)中的公钥进行加密。

(6) 接着客户端继续发送 ChangeCipherSpec 数据包。该数据包会提示服务器,在此数据包之后的通信会采用 Pre-Master-Secret 密钥加密。

(7) 客户端发送 Finished 数据包,该数据包的内容为校验值,校验值的计算方法为:对本次 HTTPS 握手双方发送所有 SSL 握手数据包,合并计算哈希结果,并将计算得出的哈希结果利用新协商的对称密钥进行加密。若服务器能够解密该数据包,并和客户端一样对至今的全部数据包进行校验,则将自身计算的校验值与客户端发送来的校验值进行比对,如果比对的结果一致,则证明 SSL 密钥协商成功。

(8) 服务器同样发送 ChangeCipherSpec 数据包。

(9) 服务器同样发送 Finished 数据包。

(10) 服务器和客户端的 Finished 包交换完毕之后,SSL 连接就算建立完成。之后通信会受到 SSL 的保护。从此处开始进行应用层协议的通信,即发送 HTTP 请求。

(11) 应用层协议通信,即发送 HTTP 响应。

(12) 最后由客户端断开连接。断开连接时,发送 close_notify 数据包。图 5.19 做了一些省略,这步之后再发送 TCP FIN 包来关闭与 TCP 的通信。

5.2.3　从数据包来看 HTTPS 建立连接的过程

HTTPS 数据包解码,使用科来 CSNAS 捕获一个 HTTPS 会话,在 TCP 会话视图中可以清楚地看到,首先进行的是 TCP 的三次握手,而后进行的才是 SSL 握手,如图 5.20 所示。

1. ClientHello 数据包(对应 5.2.2 节的“(1)”)

ClientHello 数据包如图 5.21 所示。图中展示了客户端声明的协议版本、random 随

图 5.20　TCP 三次握手

机数、支持的加密算法。需要注意的是,“扩展长度”字段用于声明 ClientHello 数据包后面所携带选项的长度,HTTPS 除固定的数据包格式外,也可以基于 TLV(Type-Length-Value)形式携带选项,与 TCP 选项类似,图中截取的字段“扩展长度”为 403 字节。

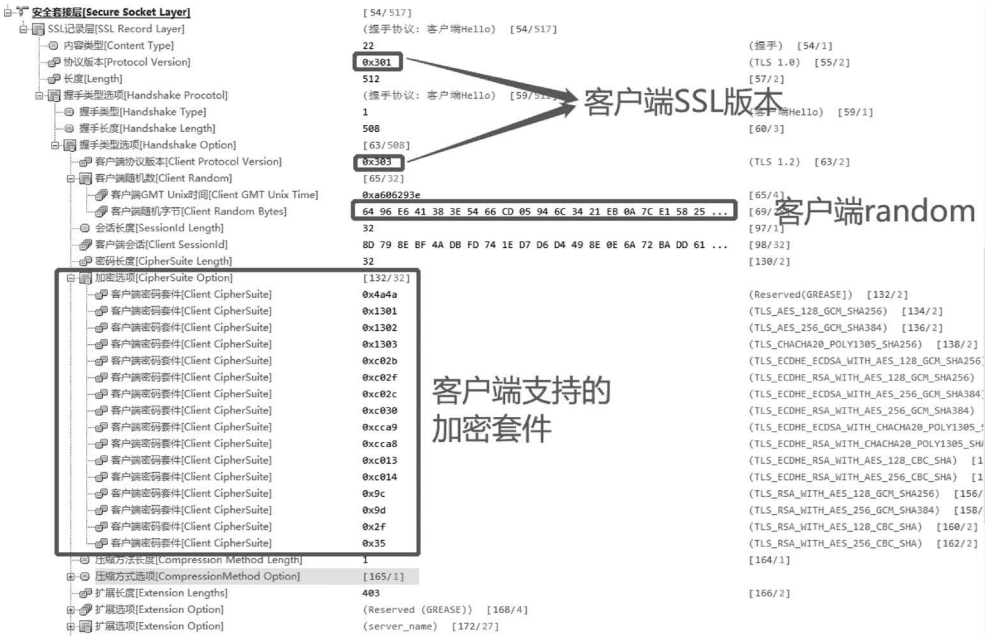


图 5.21　ClientHello 数据包解码

2. ServerHello 和 Certificate 数据包(对应 5.2.2 节的“(2)”“(3)”)

Certificate 数据包总是在 ServerHello 数据包之后立即发送,所以两项内容在同一个数据包中。如图 5.22 所示,图中的服务器选定了支持的版本,选定了密码套件,发送了 random 用于计算 Pre-Master-Secret,并且发送了公钥证书,可以通过捕获到的公钥证书数据流将公钥证书还原。

3. ServerHelloDone 和 ServerKeyExchange 数据包(对应 5.2.2 节的“(4)”)

服务器会发送握手类型 14 选项表示 ServerHello 阶段的结束。在这个阶段,如果 ServerHello 选择的加密套件为 DHE_DSS、DHE_RSA、DH_anon 等基于 DH 类的密钥传递算法,服务器会通过 ServerKeyExchange 交换 SSL 密钥中的某些数学参数,以便双方生成 Pre-Master-Secret 预主密钥(这是一个可能出现的步骤,如果不是上述三种 DH

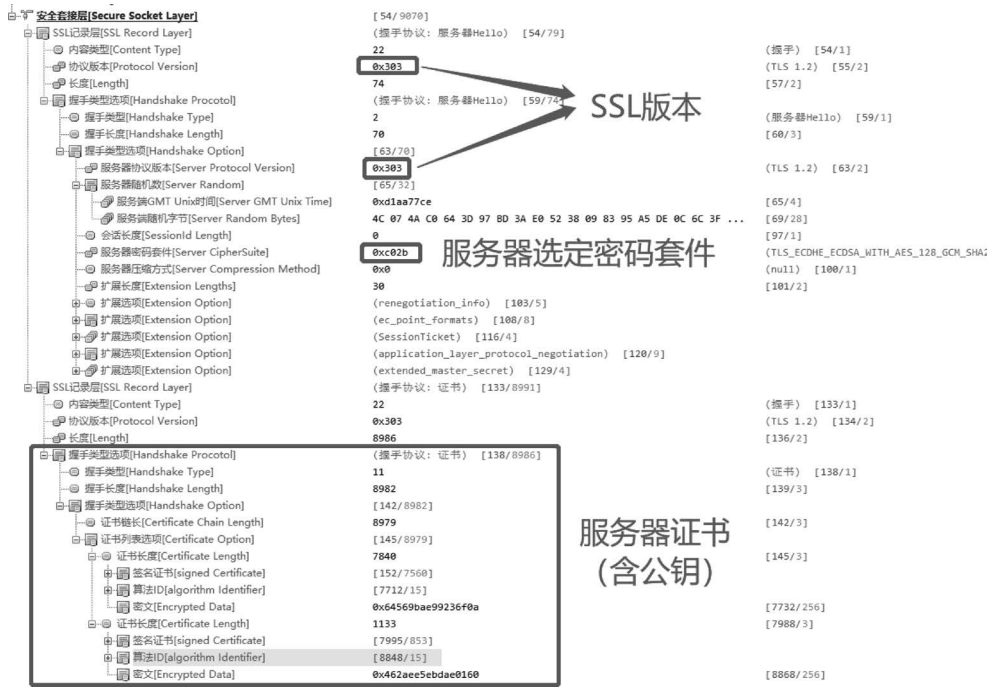


图 5.22　ServerHello 数据包解码

类算法，则不出现 ServerKeyExchange，直接发送 ServerHelloDone），分析如图 5.23 所示。

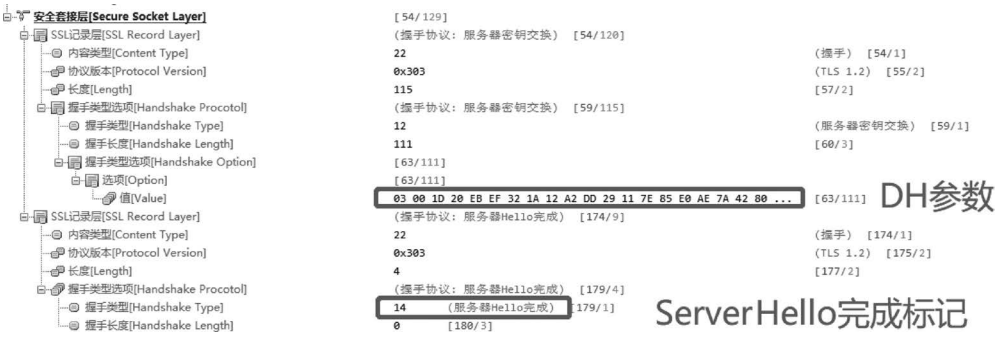


图 5.23　DH 参数与 ServerHelloDone 标记

4. ClientKeyExchange、ChangeCipherSpec 和 Finished 数据包（对应 5.2.2 节的“(5)”“(6)”“(7)”）

ClientKeyExchange 数据包用于交换 SSL 密钥中的某些数学参数，如 DH 参数，ChangeCipherSpec 数据包用于声明后续的数据包交互均启用加密，Finished 数据包用于将经过密钥加密之前的所有包进行 hash 计算，将得到的 hash 结果使用 SSL 协商的对称密钥加密，发送给服务器测试。

在实际情况中，这三种数据包可能被合并在一起发送。服务器使用 SSL 协商的对称

密钥将该加密内容解开后,与服务器自己计算的 hash 结果相对比,若对比结果一致,则 SSL 密钥协商完成,数据包分析如图 5.24 所示。

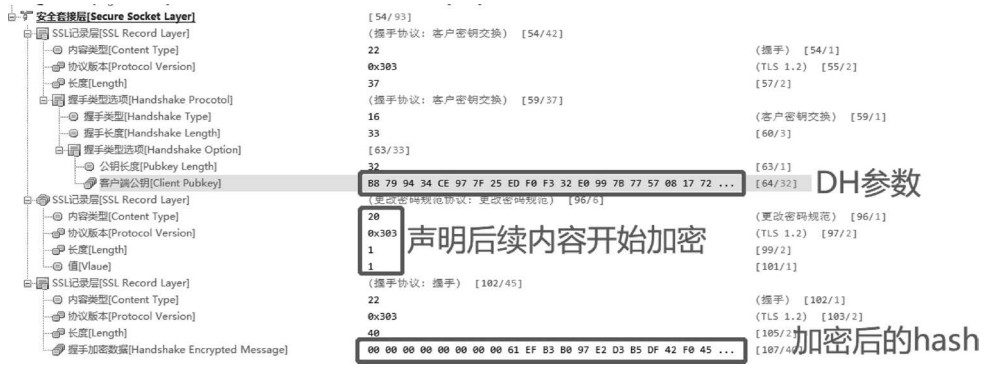


图 5.24　加密标记与加密后的数据

5. ChangeCipherSpec 和 Finished 数据包(对应 5.2.2 节的"(8)""(9)")

和客户端最终的 ChangeCipherSpec 和 Finished 数据包一样,服务器也会发送这两个数据包,分别声明后续内容开始加密,以及发送一个加密的 hash 进行测试,数据包分析如图 5.25 所示。

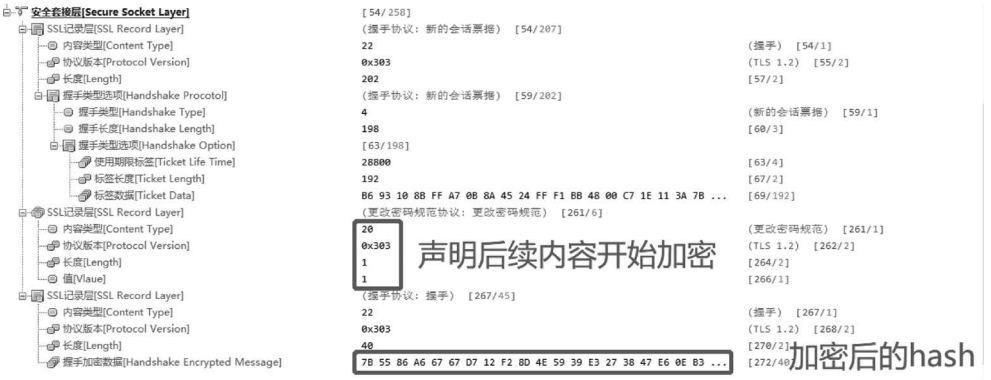


图 5.25　HTTPS 服务器启用 SSL 加密

至此,双方 SSL 建立连接成功完成。后续采用密钥加密进行 HTTP 通信,通信结束后进行正常的 4 次断开。

5.2.4　HTTPS 的分析方法

抓包 HTTP 通信能够清晰地看到通信的报头和信息的明文,但是 HTTPS 是加密通信,无法看到 HTTP 的相关报头和数据的明文信息。HTTPS 通信主要包括三个过程:TCP 建立连接、TLS 握手和 TLS 加密通信,因此主要分析 HTTPS 通信的握手建立和状态等信息。

1. 分析 ClientHello 数据包

根据 Version 信息能够知道客户端支持的最高的协议版本号,如果是 SSL 3.0 或

TLS 1.0 等低版本协议，要注意非常可能有因为版本低引起一些握手失败的情况。

2. 分析 ServerHello 数据包

根据 TLS Version 字段能够推测出服务器支持的协议的最高版本，版本不同可能造成握手失败。基于 Cipher Suite 信息可以判断出服务器优先支持的加密协议。

3. 分析 Certificate(证书)

配置服务器并返回证书链，根据证书信息并与服务器配置文件对比，对证书文件进行分析，查询会话是否被截获，或分析服务器证书是否为不安全的自签名证书。

4. 分析 Alert(告警)

告警信息会说明建立连接失败的原因，即告警类型，对于定位问题非常重要。

5.3　绰号与真名——DNS

当对网站进行访问时，通常使用 www.baidu.com 或 www.sina.com 这样的域名输入浏览器进行访问。由于 IP 的封装必定会用到 IP 地址，而非域名，因此在交互过程中，需要将访问的域名转换成对应的 IP 地址，然后进行 IP 通信。

如何将每一个需要用到的网站 IP 地址都对应成一个便于记忆的域名？如何保证互联网上所有的域名没有冲突？因此需要一个庞大的域名管理体系，来对互联网上的这些域名进行统一管理。

DNS 协议很好地解决了上述问题，并且在互联网上广泛应用，用户通过 DNS 协议能够将便于记忆的域名转换成难以记忆的 IP 地址，也可以为某些不便于输入的域名设置一个便于记忆的域名别名。譬如 www.baidu.com 是一个别名，而其真实域名是 www.a.shifen.com，别名类似于生活中的绰号，真名类似于生活中的大名，IP 地址类似于生活中的人，DNS 协议将这些绰号、真名、人进行了很好的管理与关联，并且让全世界的人都不会出现同名同姓的情况。本节来介绍 DNS 是如何进行工作的。

5.3.1　DNS 的概念

识别主机(服务器)有主机名(域名)和 IP 地址两种方式。

主机名(域名)便于记忆(如 www.colasoft.com)，但路由器很难处理(数据在网络中主要靠路由器来转发，路由器按 IP 地址寻址要方便很多)；IP 地址定长，有层次结构，便于路由器处理，但人们却难以记忆。折中的办法就是建立 IP 地址与主机名间的映射，这就是域名系统 DNS 做的工作。

DNS(Domain Name System，域名系统)是万维网上作为将域名和 IP 地址相互映射的一个分布式数据库，通常为其他应用层协议提供服务(如 HTTP、SMTP、FTP)，将主机名(域名)解析为 IP 地址。DNS 通常在四层使用 UDP 的 53 号端口(部分功能使用 TCP 的 53 号端口)进行传输。每当客户端访问某域名时，都必须先使用 DNS 协议将域名解析为 IP 地址，再通过 IP 地址访问，如图 5.26 所示。

图 5.26 中的 Client 端希望访问 www.baidu.com，此时第一个步骤并不是发送 HTTP 请求，而是向 DNS 服务器发送域名解析的请求，DNS 服务器收到请求后进行应

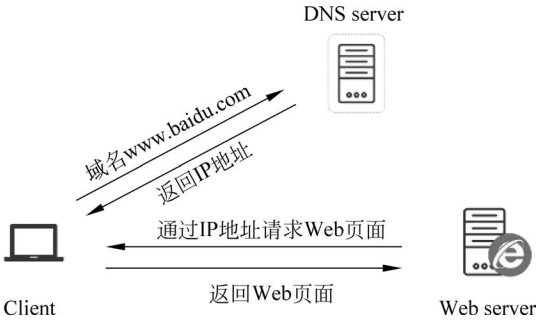


图 5.26 域名解析示意图

答,返回一个 IP 地址给 Client 端。Client 端通过得到的 IP 地址向 www.baidu.com 的服务器发送 HTTP 请求。

除了用于主机名到 IP 地址的转换外,DNS 通常还提供主机名到以下几项的转换服务:

(1) 主机命名。有着复杂规范主机名的主机可能有一个或多个别名,通常规范主机名较复杂,而别名更容易让人记忆。应用程序可以调用 DNS 来获得主机别名对应的规范主机名,以及主机的 IP 地址。

(2) 邮件服务器别名。DNS 也能完成邮件服务器别名到其规范主机名以及 IP 地址的转换。

(3) 负载均衡。DNS 可用于冗余的服务器之间进行负载均衡。一个繁忙的站点(如 abc.com)可能被冗余部署在多台具有不同 IP 的服务器上。在该情况下,在 DNS 数据库中,该主机名可能对应着一个 IP 集合,但应用程序调用 DNS 来获取该主机名对应的 IP 时,DNS 通过某种算法从该主机名对应的 IP 集合中挑选出某一 IP 进行响应。

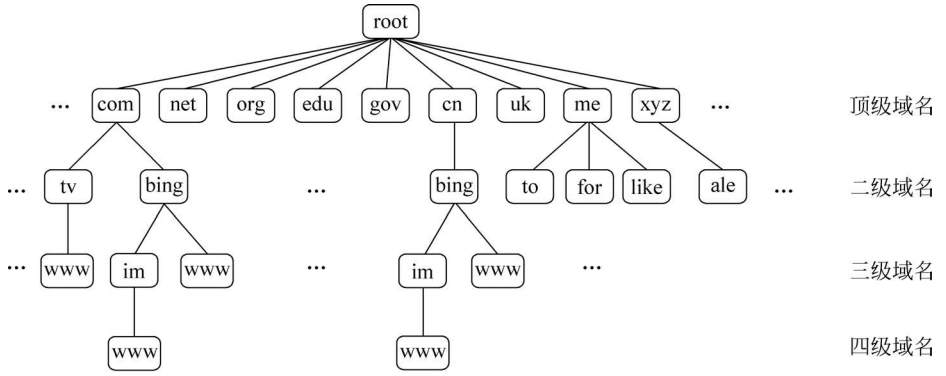
5.3.2 树状域名结构

把 DNS 的工作比作手机通讯录是一个比较合适的比喻,但域名系统并不像电话号码通讯录那么简单,由于通讯录主要是个人在使用,同一个称呼(例如父亲、母亲、儿子、女儿)可以保存在所有人的通讯录里,对应的电话号码各不相同,并无不妥,但如果让全世界使用统一的通讯录,上述称呼便不适合出现了。域名是全世界、所有人都在统一使用的,因此必须要保持唯一性。

为了让域名保持唯一性,互联网在命名的时候采用了树形结构的命名方法。该方法保证了域名的唯一性。域名的层次结构如图 5.27 所示。

图 5.27 展示了域名的层次结构,将这些域名进行排列,级别低的写在左边,级别高的写在右边,使用点号“.”分隔。例如,com 为顶级域名,colasoft 为二级域名,www 为三级域名,排列的结果为 www.colasoft.com。

树形的结构命名方法可以确保网络中的域名便于管理,不易重复,具备唯一性。但随之而来的问题是庞大繁杂的域名该由哪家机构进行管理,是由一家统一的机构全部管理,还是分别管理各自的服务器?如果分别管理,如何在分别管理的基础上避免域名重复?



域名服务主要是基于UDP实现的，服务器的端口号为53。

图 5.27　树形结构命名

因此，只有域名结构还不行，还需要有一个机制去解析域名，域名需要域名服务器去解析，域名服务器实际上就是装有域名解析系统服务端程序的计算机。按照树形结构中的级别，域名解析服务器被分为根域名服务器、顶级域名服务器、权限域名服务器，如图 5.28 所示。

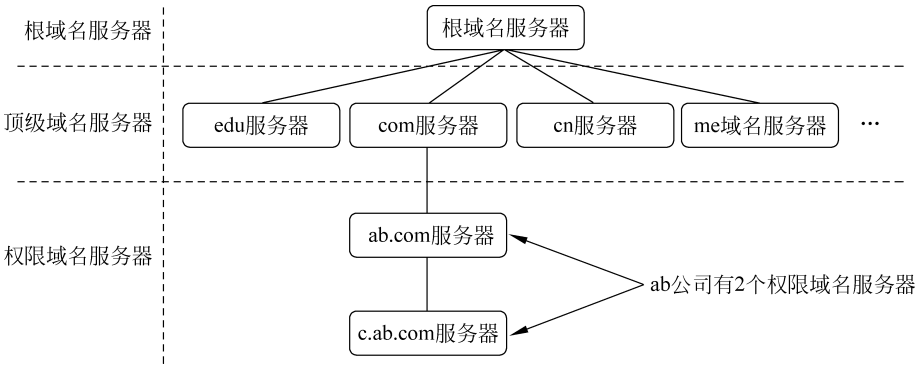


图 5.28　DNS 各级服务器

1. 根域名服务器

根域名服务器是最高层次的域名服务器，也是最重要的域名服务器，本地域名服务器如果解析不了域名，就会向根域名服务器求助。所有的根域名服务器都知道所有的顶级域名服务器的域名和地址，如果向根服务器发出对 colasoft.com 的请求，则根服务器不能在它的记录文件中找到与 colasoft.com 匹配的记录，但是它会找到 com 的顶级域名记录，并把负责 com 地址的顶级域名服务器的地址发回给请求者。

全球共有 13 台不同 IP 地址的根域名服务器，这 13 台根域名服务器的名字分别为 A～M。1 台为主根服务器在美国，其余 12 台均为辅根服务器，其中 9 台在美国，2 台在欧洲，位于英国和瑞典，1 台在亚洲，位于日本。这些服务器由各种组织控制，并由 ICANN(internet corporation for assigning names and numbers，互联网名称和数字地址分配公司)授权，由于每分钟要解析的名称数量多得令人难以置信，因此实际上每台根服

务器都有很多根服务器的镜像服务器负责帮忙分担工作,每台根服务器与它的镜像服务器共享同一个 IP 地址。当用户对某台根服务器发出请求时,请求会被转发到该根服务器离用户最近的镜像服务器上进行处理。

2. 顶级域名服务器

顶级域名服务器负责管理自己的二级域名。当根域名服务器告诉查询者顶级域名服务器的地址时,查询者紧接着就会到顶级域名服务器进行查询。比如查询 colasoft.com 时,根域名服务器返回了 com 服务器的地址,查询者会再次向 com 服务器询问顶级域名服务器 colasoft.com 的地址,顶级域名服务器进行查找,并把负责 colasoft.com 地址的二级域名服务器的地址发回给请求者。

3. 权限域名服务器

权限域名服务器往往都是二级或三级甚至四级域名服务器,这些服务器会维护自己的二级域名,例如 www.colasoft.com 是由权限域名服务器 colasoft.com 维护查询的,而 www.a.shifen.com 是由权限域名服务器 a.shifen.com 维护查询的。

5.3.3 本地域名服务器

本地域名服务器是指计算机手动配置或者自动获取的 DNS 服务器,用于为本机解析所有的 DNS 请求。如果接入家庭宽带的 WiFi 网络中,则默认本地域名服务器地址为宽带猫;如果接入大型企业网络中,则多数情况下本地域名服务器地址为企业内自建的 DNS 服务器地址;如果通过网线直连运营商网络,则很有可能本地域名服务器地址为运营商搭建的 DNS 服务器地址。Windows 操作系统的主机可以使用 ipconfig /all 命令查看本机的本地域名服务器地址,回显如图 5.29 所示。

图 5.29 DNS 配置信息

5.3.4 域名解析过程

了解了 DNS 的概念与树状域名结构后,就比较容易了解域名的解析流程了。DNS 域名解析流程简单来说与 HTTP 类似,也只有两种数据包,分别是查询请求和查询响应,使用科来 CSNAS 在本机捕获数据包,分析 DNS 域名解析查询过程,观察到的查询流程如图 5.30 所示。

过滤: protocol = dns　全部　全字匹配

编号	绝对时间	源	源地理位置	目标	目标地理位置	协议
13	17:01:24.0303...	192.168.[redacted].58:54285	本地	192.168.[redacted]252:53	本地	DNS Query
14	17:01:24.0327...	192.168.[redacted]252:53	本地	192.168.[redacted].58:54285	本地	DNS Response

图 5.30　过滤 DNS 协议的数据包

一次 DNS 解析包括以下两个步骤：

(1) 本机 192.168.x.58 向本地域名服务器 192.168.x.252 发出一个 DNS 查询请求包，里面包含请求解析的域名。

(2) 本地域名服务器向本机回应一个 DNS 查询响应包，里面包含域名对应的 IP 地址。

上述结果是在本地抓包看到的结果，是从客户端角度出发看 DNS 协议的过程，因此只能看到客户端发出 DNS 请求和接收到 DNS 应答两个步骤，实际上查询过程中还有很多由本地域名服务器去完成的步骤，如图 5.31 所示。

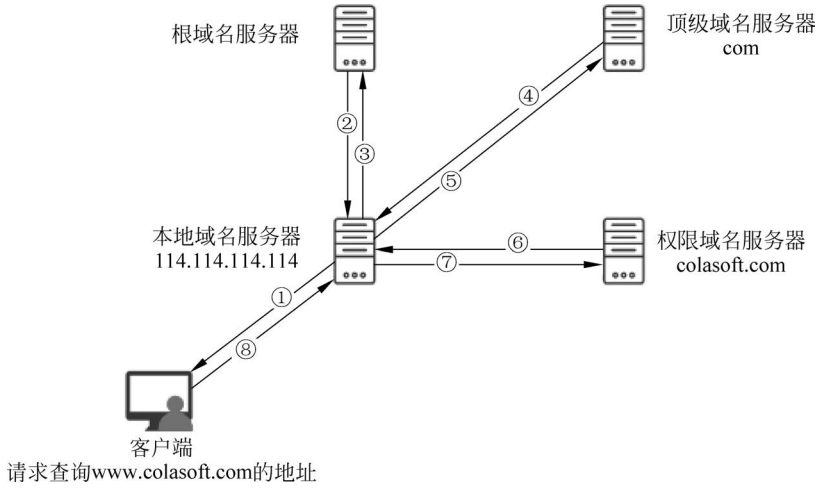


图 5.31　DNS 解析流程

请求解析流程如下：

(1) 客户端先向本地域名服务器请求查询 www.colasoft.com 对应的 IP 地址。

(2) 本地域名服务器未缓存 www.colasoft.com 对应的 IP 地址，于是向根域名服务器查询 com 域的顶级域名服务器地址。

(3) 根域名服务器告诉本地域名服务器，顶级域名 com 对应的服务器的 IP 地址。

(4) 本地域名服务器向顶级域名服务器 com 进行查询，查询 colasoft.com 域名的权限域名服务器地址。

(5) 顶级域名服务器 com 告诉本地域名服务器，colasoft.com 权限域名服务器的 IP 地址。

(6) 本地域名服务器向权限域名服务器 colasoft.com 进行查询，查询 www.colasoft.com 域名对应的 IP 地址。

(7) colasoft.com 的权限域名服务器告诉本地域名服务器 www.colasoft.com 主机

的 IP 地址。

(8) 本地域名服务器最后把得到的 www.colasoft.com 对应的 IP 地址告诉主机 A。

通过以上步骤不难看出,实际上当对一个三级域名进行 DNS 解析时,假设本地域名服务器没有缓存 www.colasoft.com 的域名,则会经过如上所述的 8 个步骤,如果有缓存,则对应的步骤可能减少。在客户端主机进行抓包时,仅能观察到步骤①～步骤⑧,这是由于抓包的位置导致的,也是由于本地域名服务器帮客户端"包办"了其余的域名查询工作,因此从客户端的角度看来,看起来 DNS 的查询与响应仅是一问一答的过程,实则本地域名服务器在背后帮客户端承担了很多工作。

为什么本地域名服务器会帮忙"包办",而不让客户端自己分别找根域名服务器、顶级域名服务器、权限域名服务器去一步一步自行请求解析呢?这是在协议设计时考虑到了节约重复的问题,因为如果网络中有 100 个客户端同时请求 www.colasoft.com,而没有本地域名服务器帮忙"包办",则这 100 个客户端各自走流程,将产生至少 3×100 次请求和 3×100 次应答,十分消耗网络的带宽。而本地域名服务器是支持缓存功能的,客户端在请求过一个域名以后,本地域名服务器会将该域名对应的 IP 地址进行缓存,这样在第二个客户端来请求解析同一个域名时,若缓存没有过期,则直接从缓存进行响应。如此,当网络中有 100 个客户端同时请求时,十分节约网络带宽。

如果本地域名服务器没有缓存相关条目,才会按图 5.31 的流程进行查询。每次收到响应信息后,都会将响应信息缓存起来。

实际上,DNS 域名解析查询的方式有以下两种:

(1) 递归查询。递归查询是"包办"式查询,客户端查询发往上级 DNS,如果上级 DNS 没有解析结果,则由上级 DNS 以自己的身份作为客户端,再继续向上级 DNS 发起递归查询,而不是让主机自己进行下一步查询。

(2) 迭代查询。迭代查询是"亲力亲为"式查询,客户端查询直接发往根域名服务器,然后查询顶级域名服务器,最后查询权限域名服务器,逐级亲自查询。

观察图 5.31 中的 DNS 解析流程,如何区分图 5.31 的解析是递归还是迭代?其实图中的步骤并不全是递归或迭代,而是同时组合了两种方法。一般情况下,客户端向本地域名服务器的查询都是采用递归查询,本地域名服务器向根域名服务器的查询都是采用迭代查询。因此,图 5.31 中的①和⑧为递归查询,②③④⑤⑥为迭代查询。

RFC1034 文档明确说明应尽量避免递归查询,在 DNS 协议中有个 RA 字段叫是否允许递归,置 0 表示不允许,置 1 表示允许。为什么在 RFC 文档中声明避免递归查询的同时,RA 一般都会置 1?其实这里有一个误区,RA 位其实不是规定客户端用什么方法去查询,而是客户端用于声明自己支持递归查询。默认方式下,客户端与本地服务器间的交互为递归,本地服务器与外部 DNS 服务器交互为迭代。

在上面的解析流程的前面其实还有一步,操作系统在发出 DNS 解析之前会先看本地是否设置了相关域名的解析,如果系统内设置了的话,会省去 DNS 请求的步骤,直接使用设置好的地址,会节省很多时间。该文件保存在 C:\Windows\System32\drivers\etc\hosts 中,文件保存了本地 DNS 解析的相关内容。

对于不同的查询请求,可能会查询不同的内容:某些查询希望获得域名对应的 IP 地

址，对应的查询类型为 A 记录；某些查询希望获得域名对应的邮件服务器地址，对应的查询类型为 MX 记录；某些查询希望获得该域名对应的起始授权服务器地址，对应的查询类型为 SOA 记录等。表 5.5 展示了 DNS 在进行查询时所希望获得的查询类型。

表 5.5　DNS 查询类型

类　型	助 记 符	说　明
1	A	由域名获得 IPv4 地址
2	NS	查询域名服务器
5	CNAME	查询规范名称
6	SOA	原始授权
11	WKS	熟知服务
12	PTR	把 IP 地址转换成域名
13	HINFO	主机信息
15	MX	邮件交换
28	AAAA	由域名获得 IPv6 地址
252	AXFR	传送整个区的请求
255	ANY	对所有记录的请求

5.3.5　DNS 数据包格式

与 TCP/UDP 等协议类似，DNS 数据包也有特定的格式，DNS 数据包格式分为 3 个部分，分别是基础结构部分、问题部分和资源记录部分（答案部分），如图 5.32 所示。

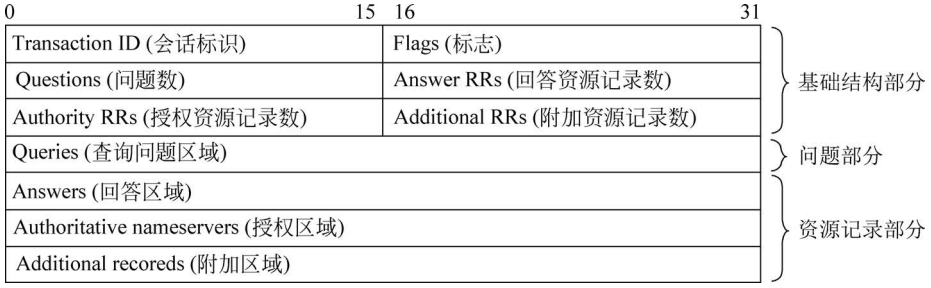


图 5.32　DNS 数据包格式

下面对 DNS 数据包中各部分的含义逐个进行介绍。

1. 基础结构部分

基础结构部分声明了 DNS 通信的一些基本需要，如通过会话标识来说明 DNS 请求与 DNS 应答包的对应关系，使用标志说明字段说明 DNS 的解析是否支持 RA 等功能，以及问题的数量和答案的数量。基础结构部分的格式是固定长度的，共 12 字节，结构如图 5.33 所示。

其中，每个部分详细介绍如下。

（1）会话标识（Transaction ID）：长 2 字节，DNS 包的 ID 标识，对于同一组请求与应答，两个数据包的会话标识应是一致的，用于表示请求包与应答包的关联。

0 15	16 31
Transaction ID (会话标识)	Flags (标志)
Questions (问题数)	Answer RRs (回答资源记录数)
Authority RRs (授权资源记录数)	Additional RRs (附加资源记录数)

图 5.33 DNS 数据包格式-基础结构部分

(2) 标志(Flags):长 2 字节,DNS 包中的标志字段,用于声明这个 DNS 包是请求/响应、标准查询/反向查询以及查询是否出现差错等,各标志的功能将在后文详细介绍。

(3) 问题数(Questions):长 2 字节,DNS 查询请求的数目,即客户端询问域名或 IP 地址的数目。

(4) 回答资源记录数(Answer RRs):长 2 字节,DNS 响应的数目,即服务器查询到的针对问题的域名对应的 IP 地址或 IP 地址对应的域名的数目,或邮件交换记录(MX)的资源记录数。

(5) 授权资源记录数(Authority RRs):长 2 字节,即在进行迭代查询时,服务器返回的下级权限名称服务器(NS)或者起始授权服务器(SOA)地址的数目。

(6) 附加资源记录数(Additional RRs):长 2 字节,额外的记录数目,如当回答区域是 MX 或者 NS 记录时,则会附加一些可能会在将来使用到的 MX 域名或 NS 域名对应的 A 记录。

在基础结构部分,标志字段一共占用了 16 位,每个字段的详细分布情况如图 5.34 所示。

0	1 4	5	6	7	8	9 11	12 15
QR	Opcode	AA	TC	RD	RA	Z	Rcode

图 5.34 DNS 标志字段格式

标志字段中的每个字段详细介绍如下。

(1) QR(1 位):查询/响应标志,0 为查询,1 为响应。

(2) Opcode(4 位):0 表示标准查询,1 表示反向查询,2 表示服务器状态查询。

(3) AA(1 位):表示授权回答。

(4) TC(1 位):表示可截断的。

(5) RD(1 位):表示期望递归。

(6) RA(1 位):表示可用递归。

(7) Z(3 位):保留字段。

(8) Rcode(4 位):表示返回码,0 表示没有差错,3 表示名字差错,2 表示服务器错误(Server Failure),1 表示格式错误,4 表示不支持所请求的查询类型。

使用科来 CSNAS 捕获到的 DNS 请求包基础结构部分如图 5.35 所示。

2. 问题部分

问题部分用来承载大多数查询中的“问题”,也就是定义所问内容的参数。查询的问题有几个,这里就有几个条目(通常为 1),每个条目的格式如图 5.36 所示。

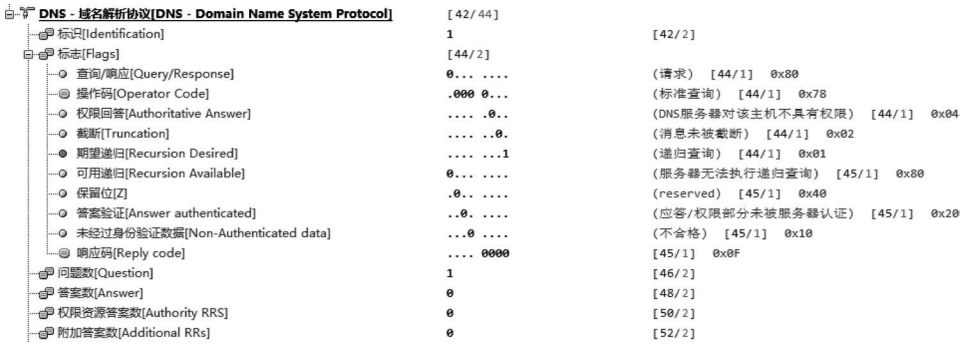


图 5.35　DNS 请求包解码

问题部分的每个字段详细介绍如下。

QNAME (查询域名)
QTYPE (查询类型)
QCLASS (查询类)

图 5.36　DNS 数据包格式-问题部分

(1) 查询域名(Name)：要查询的域名，例如 colasoft.com。

(2) 查询类型(Type)：要查询的记录类型，例如 A 记录、MX 记录、SOA 记录等。

(3) 查询类(Class)：通常为 1，表明是互联网数据。

使用科来 CSNAS 捕获到的 DNS 包问题部分如图 5.37 所示。

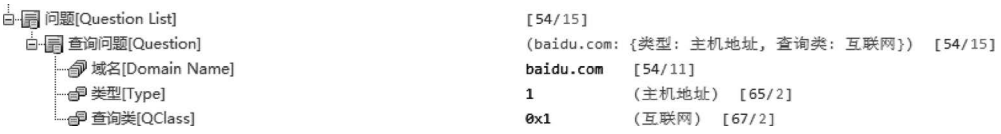


图 5.37　DNS 数据包解码-问题部分

3. 资源记录部分

资源记录部分承载了 DNS 应答中的答案，包括回答资源记录、授权资源记录和附加资源记录三种。不论响应的信息应归类于哪一种资源记录，它们的格式都是一致的，如图 5.38 所示。

Domain Name (域名，2字节或长度不固定)	
Type (查询类型)	Class (查询类)
Time To Live (生存时间)	
Data Length (数据长度)	9001
Resources Data (资源数据，长度不固定)	

图 5.38　DNS 数据包格式-资源记录部分

资源记录部分的每个字段详细介绍如下。

(1) Domain Name(域名)：查询的域名，与问题部分相同。

(2) Type(类型):查询的类型,与问题部分相同。

(3) Class(类):查询的分类,与问题部分相同。

(4) TTL(Time To Live,生存时间):TTL 字段定义了资源记录的有效时段(以 s 为单位)。该字段一般用于当地解析程序去除资源记录后决定保存及使用缓存数据的时间。

(5) Data Length(数据长度):该字段是一个 16 位的数值,其用途是给出资源数据的长度(以字节为单位),存储在任何资源记录中的数据不应大于 65 535 字节。

(6) ResourceData(资源数据):该字段的长度和内容与资源记录类型的字段有关。如果查询的是 A 记录,则这里声明的就是 A 记录对应的 IP 地址;如果查询的是其他记录,则这里声明的就是其他记录对应的答案。

使用科来 CSNAS 捕获到的 DNS 包的资源记录部分如图 5.39 所示。

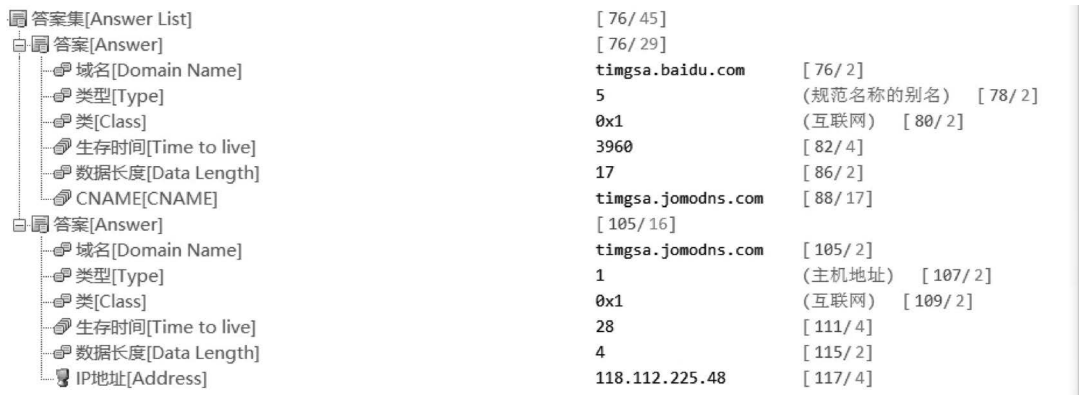


图 5.39 DNS 数据包解码-资源记录部分

在日常生活中,DNS 协议对于用户来说是透明的,但对于流量分析来说,DNS 协议的知识点还是有些枯燥和复杂,读者需要大致理解每个字段的含义,才能顺利分析 DNS 协议的数据包。

5.3.6 DNS 分析方法

分析 DNS 时基本不会去分析 DNS 解析成功与否、响应速度如何等参数,因为公网上的 DNS 服务器太多,如果某个 DNS 故障,则换另一个能够正常使用的 DNS 服务器即可,更何况 DNS 服务器不是那么容易出现故障的。因此,DNS 协议的分析更趋向于多数是安全层面的分析,例如访问了哪些域名、是否进行了 DNSLog、是否存在 DNS 放大攻击等。

科来 CSNAS 提供两种分析 DNS 流量的方式,分别是数据包解码和 DNS 日志。可以按需选择 DNS 数据包分析的方法,也可以按需将多种分析方法组合使用。

1. 数据包解码

在“数据包”视图直接观察 DNS 请求、应答包的解码,该方式比较适合详细分析 DNS 协议数据包,在解码界面中,科来 CSNAS 会解释 DNS 请求、应答包中的各个字段。

DNS 请求数据包解析如图 5.40 所示。

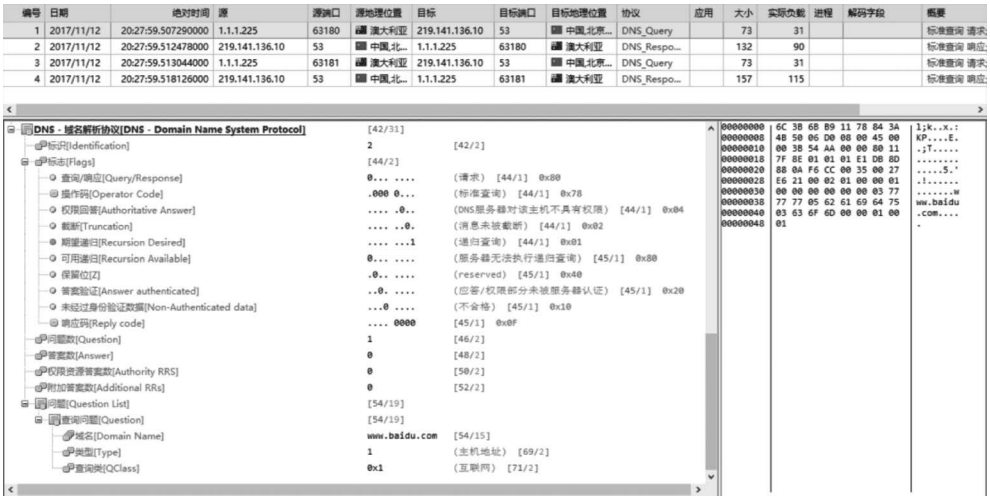


图 5.40　在“数据包”视图分析 DNS 请求数据包

DNS 应答数据包解析如图 5.41 所示。

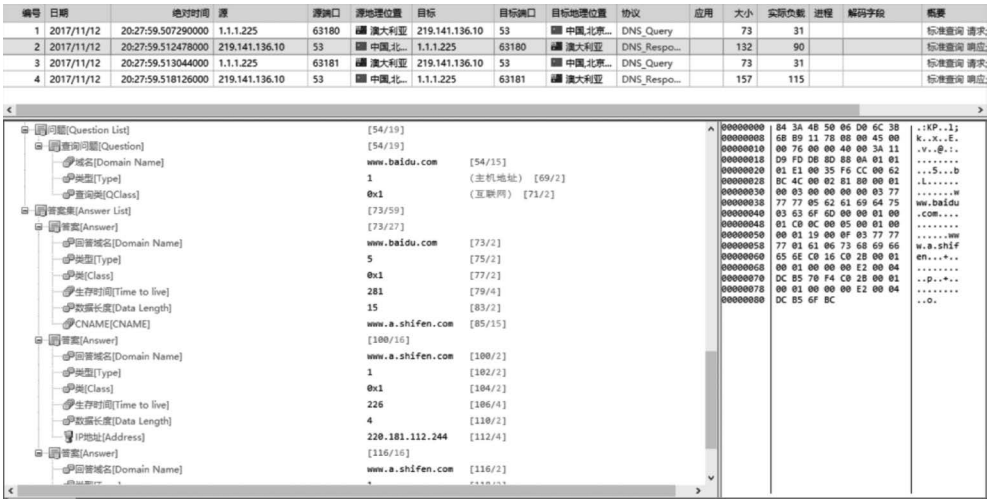


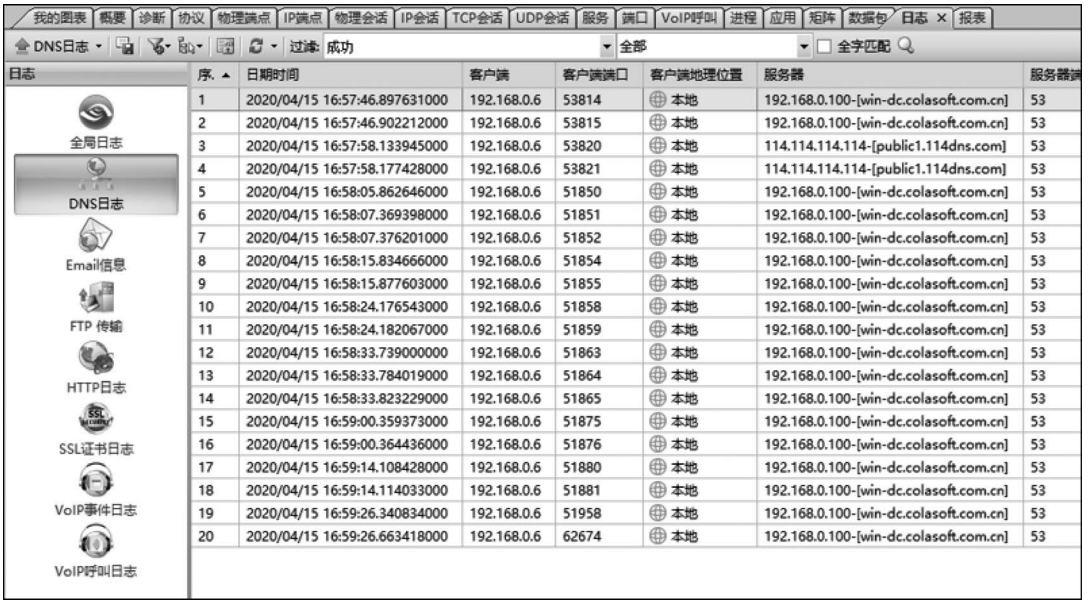
图 5.41　在“数据包”视图分析 DNS 应答数据包

与 HTTP 类似，要在数据包中过滤 DNS 协议数据包，可以使用 protocol＝dns 和 port＝53 这种语句。

2. DNS 日志

通常网络中会出现大量的 DNS 数据包，当希望在大量数据包中搜寻某个 DNS 地址时，可以使用 DNS 日志功能。在“日志”视图选中左侧的“DNS 日志”观察 DNS 日志，这些日志全部是基于捕获到的数据包生成的，并以列的形式展示，每一列为一个 DNS 请求/应答数据包，用户可以基于 DNS 日志功能快速浏览网络中出现的 DNS 行为。当日志数量过多时，DNS 日志支持以关键字作为条件进行搜索，快速列出某些特定的 DNS 请求

(如成功请求),如图 5.42 所示。



序.	日期时间	客户端	客户端端口	客户端地理位置	服务器	服务器端
1	2020/04/15 16:57:46.897631000	192.168.0.6	53814	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
2	2020/04/15 16:57:46.902212000	192.168.0.6	53815	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
3	2020/04/15 16:57:58.133945000	192.168.0.6	53820	本地	114.114.114.114-[public1.114dns.com]	53
4	2020/04/15 16:57:58.177428000	192.168.0.6	53821	本地	114.114.114.114-[public1.114dns.com]	53
5	2020/04/15 16:58:05.862646000	192.168.0.6	51850	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
6	2020/04/15 16:58:07.369398000	192.168.0.6	51851	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
7	2020/04/15 16:58:07.376201000	192.168.0.6	51852	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
8	2020/04/15 16:58:15.834666000	192.168.0.6	51854	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
9	2020/04/15 16:58:15.877603000	192.168.0.6	51855	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
10	2020/04/15 16:58:24.176543000	192.168.0.6	51858	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
11	2020/04/15 16:58:24.182067000	192.168.0.6	51859	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
12	2020/04/15 16:58:33.739000000	192.168.0.6	51863	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
13	2020/04/15 16:58:33.784019000	192.168.0.6	51864	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
14	2020/04/15 16:58:33.823229000	192.168.0.6	51865	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
15	2020/04/15 16:59:00.359373000	192.168.0.6	51875	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
16	2020/04/15 16:59:00.364436000	192.168.0.6	51876	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
17	2020/04/15 16:59:14.108428000	192.168.0.6	51880	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
18	2020/04/15 16:59:14.114033000	192.168.0.6	51881	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
19	2020/04/15 16:59:26.340834000	192.168.0.6	51958	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53
20	2020/04/15 16:59:26.663418000	192.168.0.6	62674	本地	192.168.0.100-[win-dc.colasoft.com.cn]	53

图 5.42 在“日志”视图分析 DNS 日志

在进行 DNS 抓包时,常用的 DNS 相关测试命令如下:

(1) 在 Windows 环境下可以使用 ipconfig/flushDNS 命令来清空 DNS 缓存。

(2) 在 Windows 环境下可以使用 ipconfig/displayDNS 命令来查看 DNS 缓存的内容。

(3) 在 Windows 环境和 Linux 环境下可以使用 nslookup 命令来查看域名对应的 IP 地址,如 nslookup www.abc.com。

(4) Linux 下的 host dig 命令与 nslookup 命令的功能类似。

5.4 网络邮差——SMTP 与 POP3

1896 年,中国邮政成立,到了 1987 年,来自我国的第一封电子邮件成功从北京市车道沟十号院发往德国西德卡尔斯鲁厄大学,电子邮件的内容为: Across the Great Wall we can reach every corner in the world,译为: 越过长城,走向世界。邮件内容如图 5.43 所示。

图 5.43 所示的这封电子邮件通过西门子 7760 大型计算机以及 SMTP(Simple Mail Transfer Protocol,简单邮件传输协议)发送出去。这标志着我国可以与世界通过电子邮件进行沟通和交流,是一封具有历史意义的邮件。

现在用户日常工作中一大部分工作内容是接收并回复电子邮件,即 E-Mail,简称为邮件。目前互联网邮件使用的协议仍然是 1982 年制定的 RFC821 文档中定义的 SMTP。该协议新版的说明是在 2008 年出版的互联网标准草案 RFC5321 中给出的,该说明的目的是向用户提供 SMTP 的完整结构和工作原理。

(Message # 50: 1532 bytes, KEEP, Forwarded)
Received: from unika1 by irau11.germany.csnet id aa21216; 20 Sep 87 17:36 MET
Received: from Peking by unika1; Sun, 20 Sep 87 16:55 (MET dst)
Date: Mon, 14 Sep 87 21:07 China Time
From: Mail Administration for China <MAIL@ze1>
To: Zorn@germany, Rotert@germany, Wacker@germany, Finken@unika1
CC: lhl@parmesan.wisc.edu, farber@udel.edu,
jennings%irlean.bitnet@germany, cic%relay.cs.net@germany, Wang@ze1,
RZLI@ze1
Subject: First Electronic Mail from China to Germany

"Ueber die Grosse Mauer erreichen wie alle Ecken der Welt"
"Across the Great Wall we can reach every corner in the world"
Dies ist die erste ELECTRONIC MAIL, die von China aus ueber Rechnerkopplung
in die internationalen Wissenschaftsnetze geschickt wird.
This is the first ELECTRONIC MAIL supposed to be sent from China into the
international scientific networks via computer interconnection between
Beijing and Karlsruhe, West Germany (using CSNET/PMDF BS2000 Version).

University of Karlsruhe	Institute for Computer Application of
-Informatik Rechnerabteilung-	State Commission of Machine Industry
(IRA)	(ICA)
Prof. Werner Zorn	Prof. Wang Yuen Fung
Michael Finken	Dr. Li Cheng Chiung
Stefan Paulisch	Qiu Lei Nan
Michael Rotert	Ruan Ren Cheng
Gerhard Wacker	Wei Bao Xian
Hans Lackner	Zhu Jiang
	Zhao Li Hua

图 5.43　来自中国的第一封邮件

5.4.1　SMTP 概述

SMTP 的目标是可靠、高效地发送邮件，它基于 C/S 架构进行工作，使用 TCP 的 25 号端口进行信息通信。当 SMTP 客户端有要发送的消息时，它将向 SMTP 服务器的 25 端口建立 TCP 会话。SMTP 客户端的职责是将邮件传输到一个或多个 SMTP 服务器，或报告其失败；SMTP 服务器的职责是将邮件从一个邮件域转发到另一个邮件域，例如从@sina.com 转发到@163.com。

SMTP 的通信过程包括如下三个步骤：

(1) 邮件的发送端(客户端)与接收端(服务器)的 25 端口建立 TCP 连接。

(2) 客户端向服务器发送操作命令，来请求各种服务(如登录、指定发件人与收件人等)。

(3) 服务器解析用户发来的操作命令，进行相应的操作并使用状态码进行应答。

【提示】 为了发送一封电子邮件，涉及多个命令操作，因此第(2)步的操作命令与第(3)步交替进行，直至邮件发送完毕，连接关闭。

5.4.2　SMTP 操作命令与状态码

客户端使用各种各样的操作命令来要求服务器进行对应的操作，如通过 RCPT 指定收件人、通过 MAIL 指定发件人、开始传送邮件等。具体的 SMTP 客户端操作命令如表 5.6 所示。

表 5.6　SMTP 客户端操作命令

操 作 命 令	说　　明
HELO 或 EHLO	TCP 连接建立之后，SMTP 就绪，此时服务器与客户端会互相问候，确认彼此的工作状态是否正常。服务器通过发送 HELO 问候收件方，后面跟的是服务器的标识(多为服务器主机名)

续表

操作命令	说　明
MAIL	用来表示开始编写邮件,后面跟随发件人的邮件地址。当邮件无法送达时,也用来发送失败通知
RCPT	用来表示邮件收件人的邮箱。当有多个收件人时,需要多次使用该命令,每次只能指明一个人
DATA	用来表示开始编写邮件正文
VRFY	用于验证指定的用户/邮箱是否存在
EXPN	验证给定的邮箱列表是否存在,扩充邮箱列表,常被禁用
HELP	查询服务器支持的命令
NOOP	这个命令不影响任何参数,仅要求接收方回应 OK 即可,不影响缓冲区内的数据
QUIT	用于关闭 SMTP 连接
RSET	用于通知收件方清空缓存,所有已写了一半的邮件,包括收件人、发件人全部清除

客户端使用操作命令来要求服务器进行对应的操作,服务器通过状态码来对客户端进行应答,在这一点上,SMTP 的服务器应答与 HTTP 的应答有相似之处,不只是状态码机制的相似,状态码的设计也有一些相似。例如,1XX 的状态码无论是在 HTTP 中还是在 SMTP 中均用于表示“肯定的初步答复”,2XX 的状态码均表示“肯定的完成答复”,但 3XX、4XX、5XX 的状态码与 HTTP 不同。实际上,HTTP 与 SMTP 仅是状态码机制相类似,内容并不相同。具体的 SMTP 服务器状态码如表 5.7 所示。

表 5.7　SMTP 服务器状态码

状　态　码	说　明
211	系统状态或系统帮助响应
214	帮助信息
220	<domain>服务就绪
221	<domain>服务关闭
250	要求的邮件操作完成
251	用户非本地,将转发向<forward-path>
334	等待用户输入验证信息
354	开始邮件输入,以“.”结束
421	<domain>服务未就绪,关闭传输信道
450	要求的邮件操作未完成,邮箱不可用
451	放弃要求的操作,处理过程中出错
452	系统存储不足,要求的操作未执行
501	参数格式错误
502	命令不可实现
503	错误的命令序列
504	命令参数不可实现
550	要求的邮件操作未完成,邮箱不可用

续表

状　态　码	说　　明
551	用户非本地，请尝试< forward-path >
552	过量的存储分配，要求的操作未执行
553	邮箱名不可用，要求的操作未执行
554	操作失败

如果对SMTP熟悉，要记住常见的SMTP的状态码是比较容易的，如想记住全部状态码，可参照如表5.8所示的SMTP状态码的设计原则，理解设计原则有助于记忆状态码。

表5.8　SMTP状态码的设计原则

第一位数字：X的含义	(1yz)肯定的初步答复
	(2yz)肯定的完成答复
	(3yz)肯定的中间答复
	(4yz)瞬间否定完成答复
	(5yz)永久否定完成答复
第二位数字：Y的含义	(x0z) 语法错误
	(x1z) 这些是对信息请求的回复，例如帮助或状态
	(x2z) 答复引用控制和数据连接
	(x3z) 身份验证和记账，对登录过程和记账过程的答复
	(x4z) 尚未指定
	(x5z) 文件系统，这些答复指示服务器文件系统相对于请求的传输或其他文件系统操作的状态

5.4.3　SMTP的工作流程

5.4.2节叙述了有关SMTP的基础理论知识，可以看出该协议的基础理论部分比DNS、HTTP等协议要简单。SMTP的交互包括4个阶段：连接建立、身份认证、命令交互和连接断开。图5.44展示了4个阶段的关系。

1. 连接建立阶段

客户端通过随机端口号连接SMTP服务器的25端口，通过三次握手建立连接。连接建立后，由服务器主动向客户端打招呼："220 你好，我是 XXXXXX"，客户端会通过HELO或EHLO回应这个招呼："EHLO 你好，我是 XXXXXX"。此时，双方回复的XXXXXX多为计算机主机名，通过对这个流程的分析，有机会获得通信双方的计算机名。连接建立示例如图5.45所示。

2. 身份认证阶段

服务器会要求客户端进行身份验证，因此服务器会发送一个开始验证的消息："250 就绪，支持的验证方式有 A/B/C"，此时客户端会通过"AUTH A/B/C"来进行应答，选择A/B/C三种验证方式中的一种来进一步进行身份验证。SMTP支持的验证方式有LOGIN、PLAIN、CRAM-MD5、NTLM等。这里由于SMTP本身并没有加密，因此如果

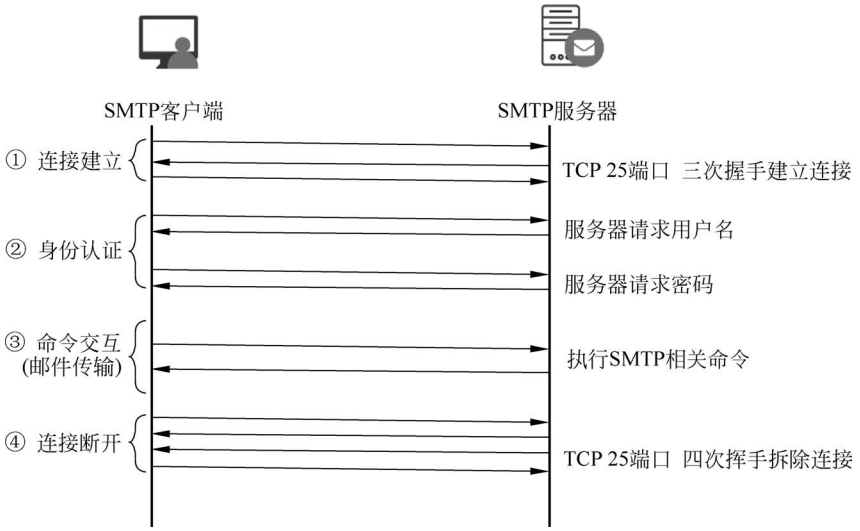


图 5.44 SMTP 的工作流程

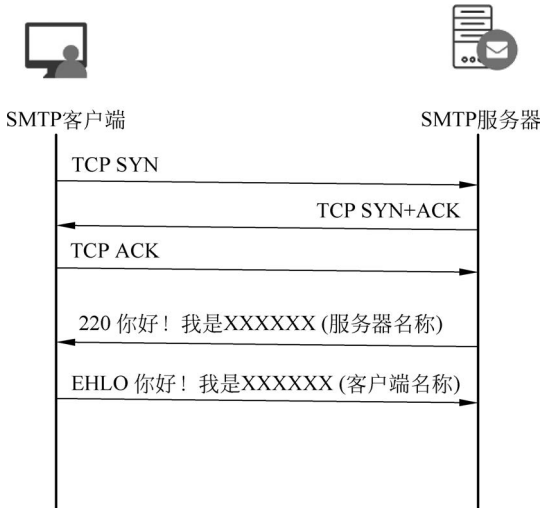


图 5.45 SMTP 连接建立

这个环节的流量被捕获到,则意味着登录的用户名和密码可以被捕获到。图 5.46 的示例中,服务器提供了 LOGIN 和 PLAIN 两种方式供客户端进行选择,客户端选择了 LOGIN 方式。在 LOGIN 方式下,由服务器分别提出“请输入用户名”和“请输入密码”的询问,客户端依次回答,提问和应答的数据流经过 Base64 编码。如果捕获到了这里的流量,可以通过 Base64 解码器对用户名和密码进行解码,用于检测邮件用户是否使用了弱口令。

如果验证通过,则服务器会返回“235 验证通过”的消息,此时身份认证阶段结束。SMTP 认证方式如图 5.46 所示。

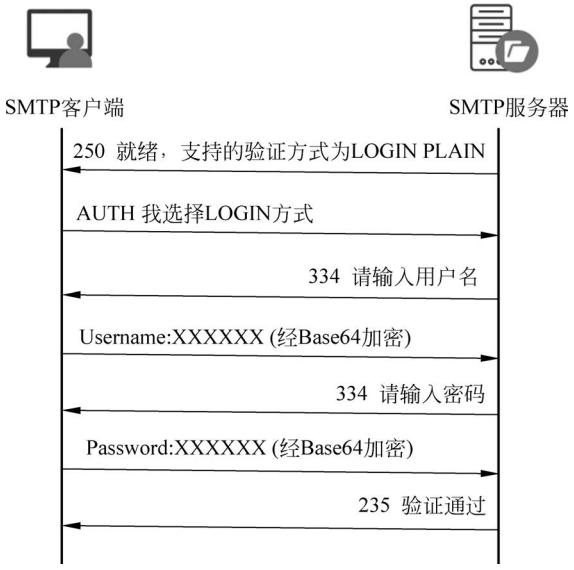


图 5.46　SMTP 认证方式

3. 命令交互阶段

命令交互阶段实现了邮件的发送功能。首先由客户端通过 MAIL 命令声明邮件的寄件人，同时声明邮件大小。服务器正确处理后，会以 250 OK 回应。应答正确后，客户端会以 RCPT 命令指定收件人的地址，如果需要指定多个收件人，则需以多次 RCPT 命令分别指定。邮件的正文部分以 DATA 命令开始，服务器返回 354 以后，客户端开始发送邮件正文，以某个特殊符号标记作为结束。而后服务器返回 250 Queued 表示邮件已经正常进入发送队列。命令交互示例如图 5.47 所示。

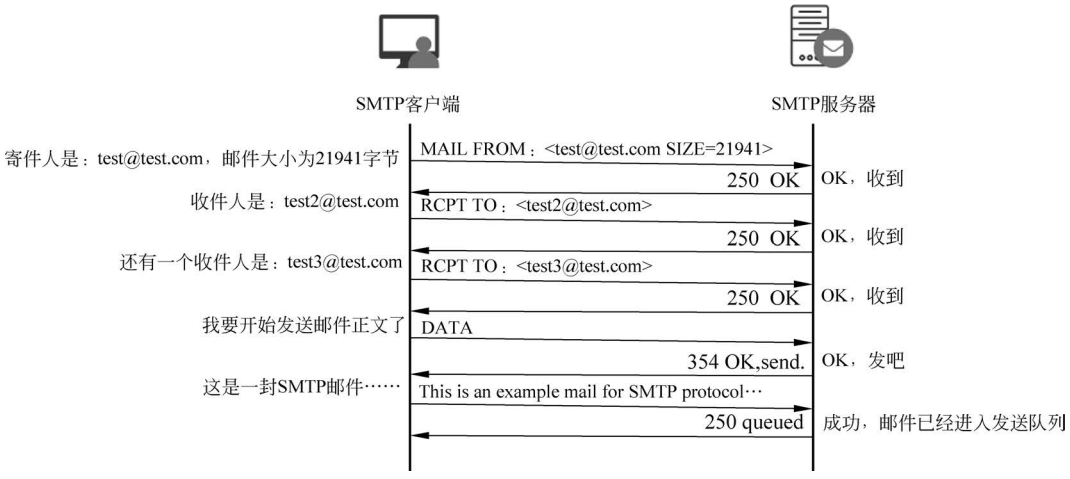


图 5.47　SMTP 命令交互过程

4. 连接断开阶段

在邮件发送结束后,进入连接断开阶段,客户端发送 QUIT 命令请求断开连接,服务器以 221 Goodbye 作为应答,然后由服务器主动 4 次挥手断开 TCP 连接。连接断开示例如图 5.48 所示。

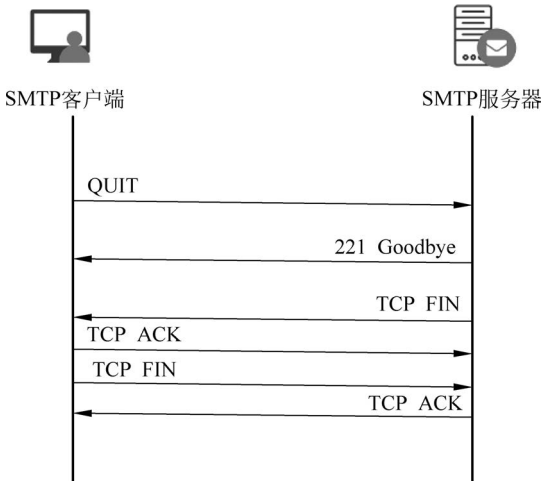


图 5.48 SMTP 连接断开阶段

至此,一次完整的 SMTP 流程就结束了。通过对 SMTP 流量的分析,可以观察到很多信息,如双方登录的主机名,服务器支持的验证方式,客户端登录使用的用户名和密码,邮件的发件人、收件人和邮件正文。因此,一些邮件安全系统宣称可以检测邮件内容或者还原邮件附件等,实际上这些产品的底层仍使用流量分析与采集技术。下面阐述如何通过流量分析还原捕获到的邮件内容。

5.4.4 POP3 概述

POP3(Post Office Protocol Version 3,邮局协议版本 3)的作用是将存储在邮件服务器上的邮件离线下载到本地。它与 SMTP 类似,也是处理邮件使用的协议,但区别在于 SMTP 用于发送邮件,POP3 用于接收邮件。POP3 使用 C/S 架构进行工作,使用 TCP 的 110 端口进行通信。

POP3 会话在生命周期中有以下三种状态。

(1) 确认状态:当客户机与服务器建立 TCP 连接时,即进入此状态。

(2) 操作状态:客户机向服务器明文发送账号和密码,服务器确认信息无误后,即进入此状态。

(3) 更新状态:在完成列出未读邮件等相应操作后,客户端发出 QUIT 命令,即进入此状态。下载未读邮件后返回"确认"状态并断开与服务器的连接。

5.4.5 POP3 操作命令与状态码

POP3 客户端使用操作命令来要求服务器进行对应的操作,服务器通过状态码来对

客户端进行应答，在这一点上，POP3 的工作流程与 SMTP 类似。POP3 的操作命令如表 5.9 所示。

表 5.9　POP3 的操作命令

操 作 命 令	说　　明
USER	用户身份确认时提供用户名
PASS	用户身份确认时提供密码
APOP	指定邮箱的字符串和 MD5
STAT	请求服务器发挥关于邮箱的统计资料，如邮件总数和总字节数
UIDL	返回邮件的唯一标识符，POP3 会话的每个标识符都是唯一的
LIST	返回邮件数量和每个邮件的大小
RETR	返回指定邮件的全部文本
DELE	服务器将由参数表示的邮件标记为删除，由 QUIT 命令执行
RSET	服务器将重置所有标记为删除的邮件，用于撤销 DELE 命令
TOP	服务器将返回由参数标识的邮件的前 n 行内容
NOOP	服务器返回一个肯定的响应
QUIT	结束会话

相比 SMTP 使用状态码作为应答，POP3 服务器返回信息的方式更加简洁，POP3 使用“＋”或者“－”表示正响应/负响应，如表 5.10 所示。

表 5.10　POP3 状态码

＋OK	正响应
－ERR	负响应

5.4.6　POP3 的工作流程

5.4.5 节叙述了有关 POP3 的基础理论知识，本节通过流量分析技术分析 POP3 交互的流程。POP3 的交互阶段与 SMTP 相同，包括 4 个阶段：连接建立、身份认证、命令交互和连接断开。图 5.49 展示了 4 个阶段的关系。

1. 连接建立阶段

客户端通过随机端口号连接 POP3 服务器的 110 端口，通过三次握手建立连接。连接建立后，由服务器主动向客户端打招呼：“＋OK 你好，我是 XXXXXX”，客户端不会对此消息进行回应，通过对这个流程的分析，有机会获得服务器双方的计算机名，信息交换如图 5.50 所示。

2. 身份认证阶段

客户端需要输入用户名和密码进行身份验证，由于不同 POP3 客户端的工作方式不同，某些客户端会通过发送 AUTH 消息询问服务器支持哪种验证方式，然后选择其中一种方式进行登录，也有些客户端会直接通过 USER 和 PASS 消息发送用户名和密码进行

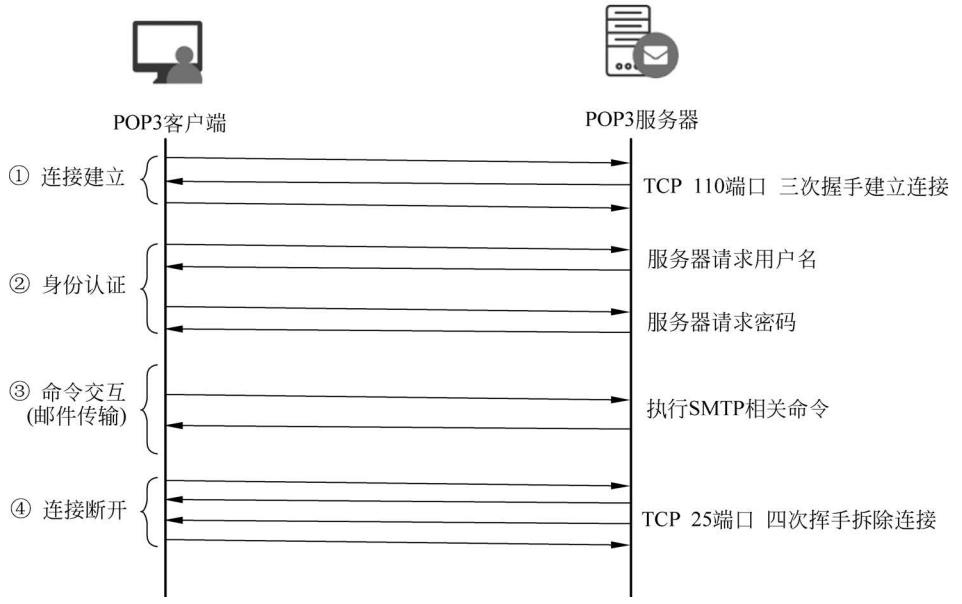


图 5.49 POP3 的工作流程

登录。在这个阶段进行登录是明文的,只要能捕获到流量,就能检测是否存在用户弱口令。身份认证阶段示例如图 5.51 所示。

图 5.50 POP3 建立连接示意图

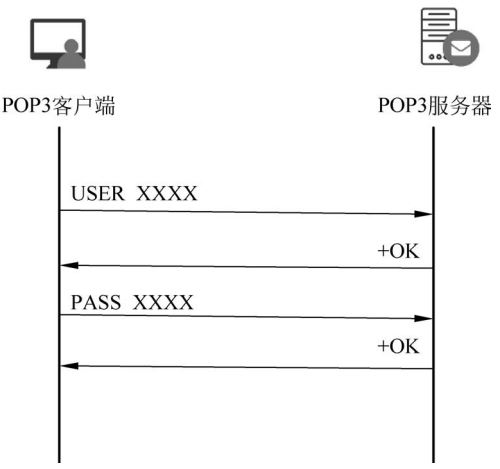


图 5.51 POP3 身份认证阶段示意图

3. 命令交互阶段

客户端一般来说会先通过 STAT 命令向服务器询问当前服务器中存储着几封邮件,服务器返回+OK 表示有两封邮件,共 2008 字节,注意这里所谓的两封包括客户端的“已读邮件”和“未读邮件”。然后客户端可以通过 LIST 命令来查询两封邮件的大小,也可以直接发送 UIDL 查看每一封邮件的 UID,邮件的 UID 是唯一的,客户通过本地已经接收

过的邮件 UID 和 POP3 服务器中的邮件 UID 进行比对，确认哪些邮件已经下载到本地，哪些没有下载。服务器返回了两封邮件的 UID 后，客户端选择将第一封邮件进行下载，通过 RETR 1 命令指定下载第一封邮件，而后服务器返回＋OK，开始将第一封邮件的内容传输给客户端。最后，客户端可以选择是否通过 DELE 命令将已经从服务器下载来的邮件从 POP3 服务器删除，如果删除，则以后无法下载，若不删除，则本地邮件清空后，仍可以重新从邮件服务器下载。命令交互阶段示例如图 5.52 所示。

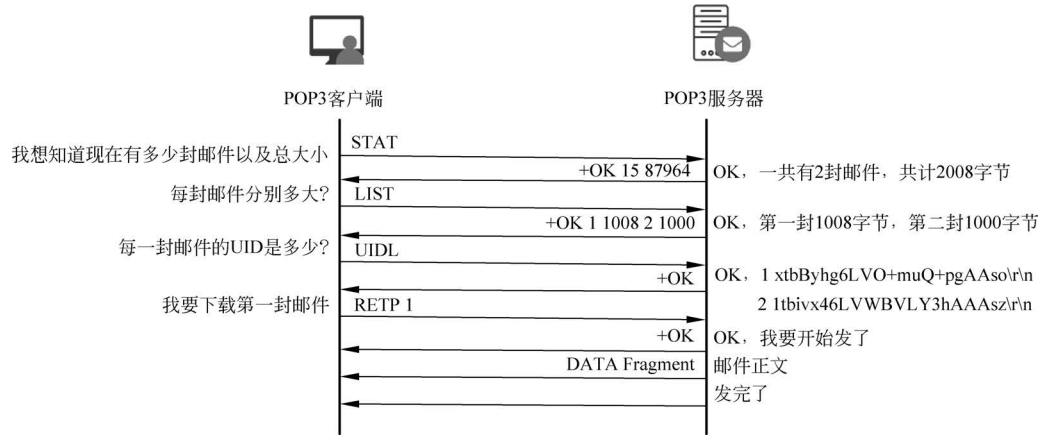


图 5.52　POP3 命令交互阶段示意图

【经验分享】 笔者曾经遇到过办公计算机不得已重新安装操作系统的窘境，系统重新安装后，所有的邮件都清空了，当笔者重新打开 Outlook 软件想要查询新邮件时，竟意外发现之前所有的邮件全都从 POP3 服务器获取了回来。这就说明笔者的计算机 Outlook 客户端没有设置“X 天后删除服务器上的邮件副本”，因此不会在从 POP3 接收邮件后，发送 DELE 消息删除服务器上的邮件副本，如图 5.53 所示。

4. 连接断开阶段

在接收邮件完毕之后，POP3 客户端通过发送 QUIT 命令表示结束会话，服务器返回＋OK 后，主动发起 TCP FIN 完成 4 次挥手断开连接。连接断开阶段示例如图 5.54 所示。

至此，一次完整的 POP3 流程就结束了。通过对 POP3 流量的分析，可以观察到很多信息，如服务器主机名、服务器软件版本、客户端登录使用的用户名和密码、邮件正文等信息。无论是分析 POP3 还是 SMTP，都可以通过流量将邮件正文进行还原，检测邮件中出现的安全隐患。

POP3 知识点总结如下：

(1) POP3 使用 TCP 的 110 端口建立连接传输邮件。

(2) POP3 客户端发送操作命令，POP3 服务器返回＋OK 或－ERR。

(3) POP3 共有 3 个状态，分别是确认状态、操作状态和更新状态。

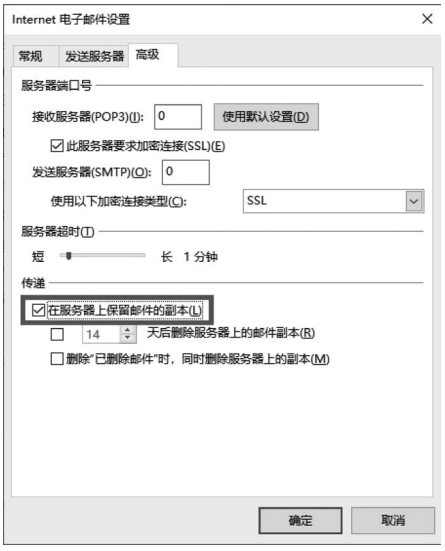


图 5.53 是否在 POP3 服务器保留邮件

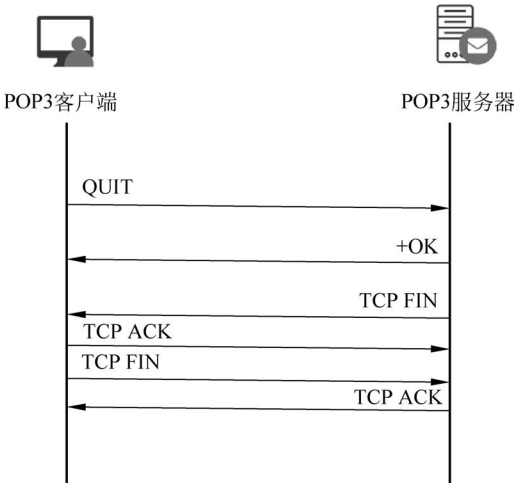


图 5.54 POP3 连接断开阶段示意图

5.4.7 邮件协议分析方法

科来 CSNAS 提供了三种分析邮件数据包的方式,分别是数据包解码、数据流解码和邮件日志。建议读者在分析邮件流量时使用数据流解码、邮件日志这两种方式对邮件流量进行分析。

【注意】 可以在计算机上使用 Outlook、Foxmail 等软件进行发送邮件的抓包分析,但请不要使用基于浏览器的网页邮件客户端,因为可能涉及 HTTPS 解密的问题。

1. 数据流解码

在"TCP 会话"视图观察 SMTP 的 TCP 会话,单击"数据流",直接观察 TCP 通信数据流,在这里按照前面章节叙述的工作流程观察 SMTP 流量,该方式能够快速分析这条 SMTP 会话的交互内容,更适合对 SMTP 流量分析的初学者,信息如图 5.55 所示。

在图 5.55 中的身份认证阶段,两条 334 的绿色消息和对应的两条蓝色客户端发送消息经过了 Base64 的编码,读者可以

图 5.55 TCP 数据流解码分析邮件流量

使用科来编解码转换工具对其进行转码，该工具通过科来 CSNAS 中的“工具”栏启动，如图 5.56 所示。

图 5.56　科来 CSNAS 内置的离线编解码转换工具

启动之后，软件默认工作在“Base64 编码/解码”视图，将 Base64 密文输入软件的“原始信息”中，单击“解码”按钮，即可在“结果：”一栏看到解码结果。图 5.57 展示了对用户名的解码结果。

图 5.57　科来 CSNAS 解码结果

按照上述方法，可以自行对图 5.57 中的 Base64 解码，观察此数据包在实验环境下的登录密码是多少。

2. 邮件日志

当数据包中出现的 Email 数量过多，不方便依次查看 TCP 会话数据流进行分析时，可以在“日志”视图选中左侧的“Email 信息”观察 Email 日志，这些日志全部是基于捕获到的数据包生成的，并以列的形式展示，每一列为一次邮件交互，用户可以使用日志功能快速浏览网络中出现的邮件行为。当日志数量过多时，日志支持以关键字作为条件进行搜索，快速列出某些特定的 Email 请求（如邮件标题），操作如图 5.58 所示。

图 5.58 以邮件日志方式分析邮件流量

当发现某条可疑的 Email 日志时,双击这条日志,即可弹出新窗口,该窗口显示这条日志对应的 TCP 会话数据流,信息如图 5.59 所示。

图 5.59 双击日志后的新窗口

5.4.8 通过 SMTP 和 POP3 还原邮件正文与附件

基于捕获到的流量,可以将 SMTP、POP3 的流量中的邮件进行还原,包括邮件中的正文、附件等信息,这对于垃圾邮件的分析十分有意义。以 SMTP 为例,操作步骤如下:

(1) 在流量中找到对应会话,打开 TCP 数据流分析界面。

通过“TCP 会话”视图,寻找 25 端口的 TCP 会话,双击打开会话分析窗口,或在 Email 日志界面双击某条日志,打开会话分析窗口,操作如图 5.60 所示。

(2) 在会话分析窗口中,单击“数据流”视图,信息如图 5.61 所示。

(3) 在“数据流”视图中,单击 OX 按钮,启动 HEX 分析界面,信息如图 5.62 所示。

(4) 在 HEX 分析界面右侧找到一串以“From:”起始的长消息,单击选中首个字符,信息如图 5.63 所示。

(5) 下拉右侧的滚动条,找到这条消息结束的地方,即对方返回的 250 消息之前,信息如图 5.64 所示。

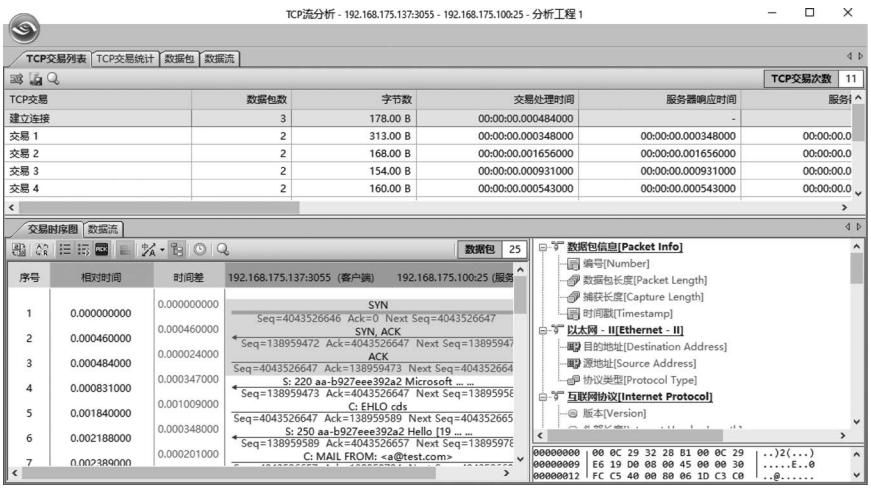


图 5.60　会话分析窗口

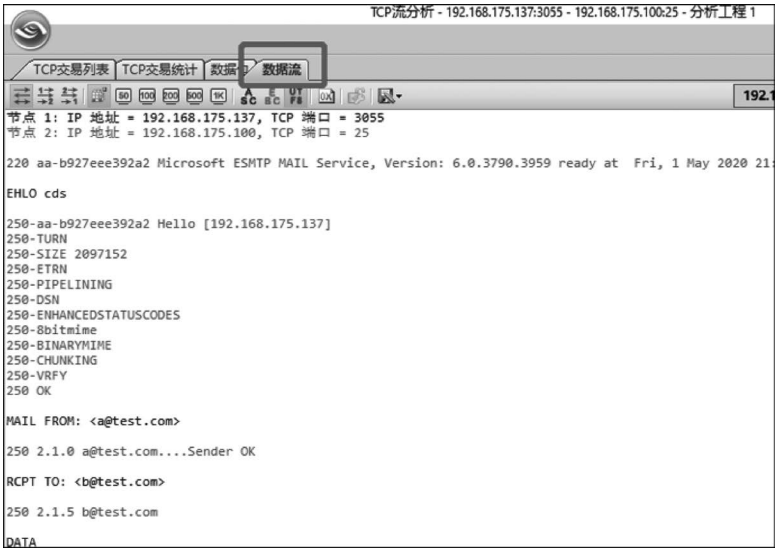


图 5.61　“数据流”视图

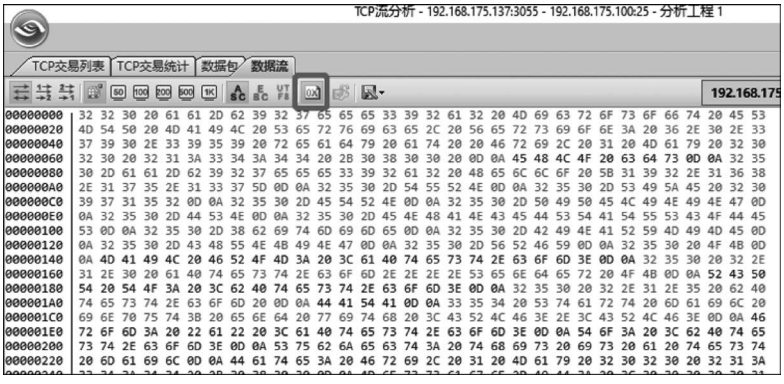


图 5.62　启动 HEX 分析

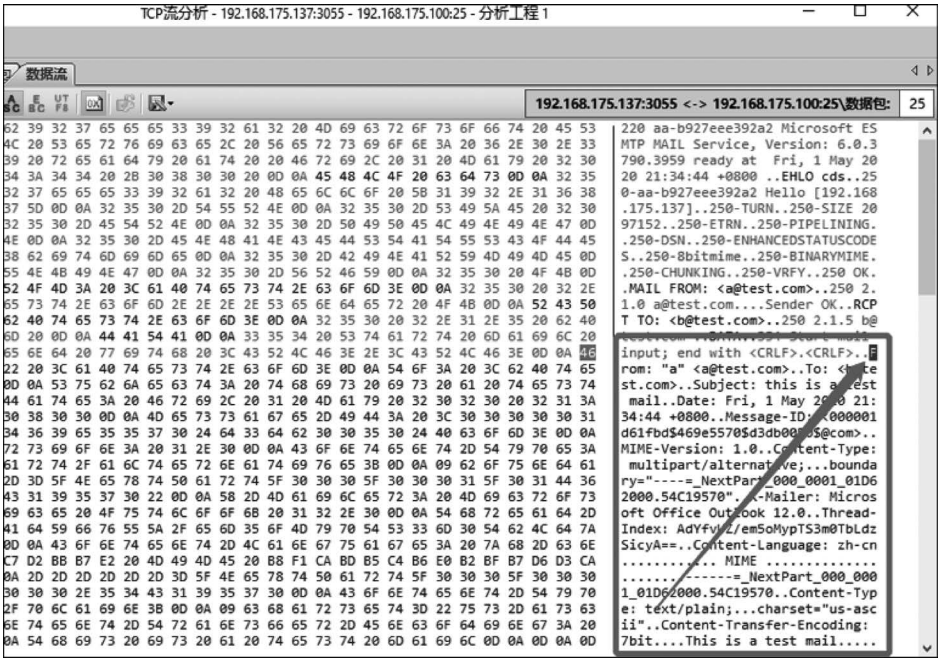


图 5.63　找到邮件消息的首个字符

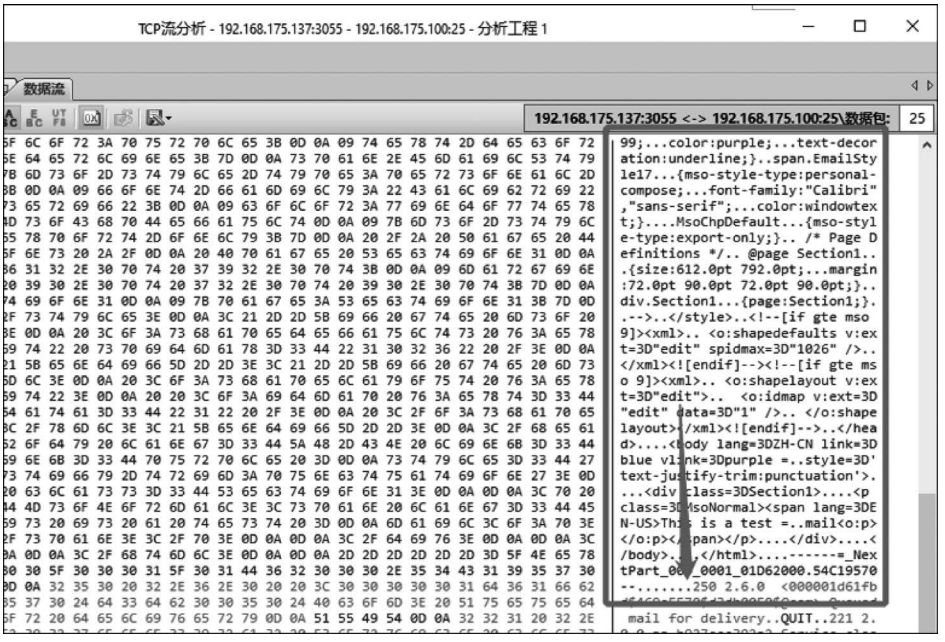


图 5.64　选中邮件消息的最后一个字符

(6) 右击“导出选择”命令,如图 5.65 所示。

(7) 将文件另存为“邮件.eml”文件,注意选择保存类型为 All Files,如图 5.66 所示。

(8) 保存文件,单击“打开文件”按钮,如图 5.67 所示。

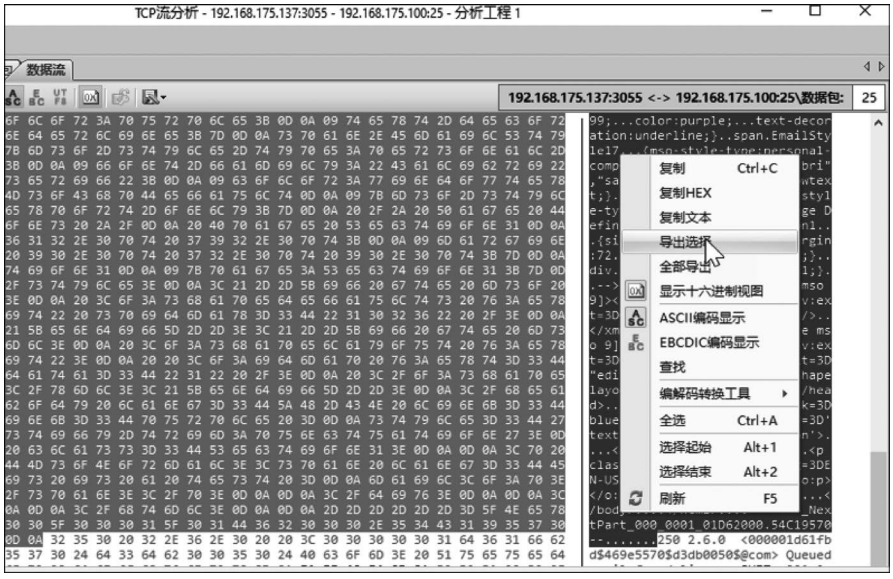


图 5.65　右击后选择“导出选择”命令

图 5.66　将 HEX 另存为.eml 邮件

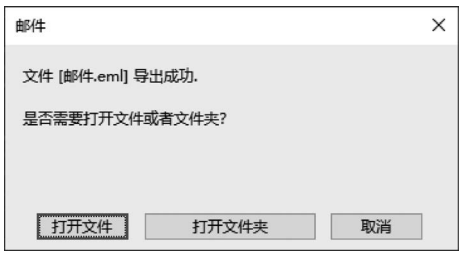


图 5.67　科来 CSNAS 打开新的邮件文件

(9) 通过 Outlook 或 Foxmail 对邮件内容进行分析,信息如图 5.68 所示。

图 5.68 打开保存的邮件

通过上述方法可以完整还原 SMTP、POP3 的邮件内容,如果对协议交互的内容足够熟悉,也可以直接通过数据流判断邮件内容,但如果要获取邮件中的附件,建议使用上述办法将邮件完整还原,这样更加有利于邮件的整体分析。

5.5 专业货运——FTP

前面介绍了网络如何通过 DNS 解析域名、如何通过 SMTP 和 POP3 收发电子邮件、如何通过 HTTP 传输超文本。但这些协议发送的内容往往是小文件,当遇到大文件的上传、下载处理时,则不适合使用 HTTP,虽然 HTTP 支持以 POST 进行上传,但上传后的文件可以被 GET 方法运行,存在一些安全隐患。FTP(File Transfer Protocol,文件传输协议)是可靠、高效的传输文件的专用协议。在文件传输方面,HTTP 与 FTP 的区别类似于生活中的小件快递与大件物流的区别,可以将 FTP 视作生活中的专业运输工。接下来,请读者随笔者一起,通过流量分析的视角进入 FTP 的货运世界。

5.5.1 FTP 概述

FTP 是专门用于传输文件的协议,它通过 TCP 的 20、21 端口进行信息传递。FTP 是一个典型的多通道协议,其中 21 端口用于建立“控制连接”,该连接用于传输用户名、密码、上传、下载等操作命令,20 端口用于建立“数据连接”,该连接用于传输上传或下载的文件。

FTP 的通信过程如下:

(1) FTP 的客户端与服务器的 21、20 端口建立 TCP 连接,其中 21 端口为发送指令使用的数据连接,20 端口为传输数据使用的数据连接。

(2) 客户端向服务器发送操作命令,来请求各种服务(如登录、打印目录、下载文件等)。

(3) 服务器解析用户发来的操作命令,进行相应的操作并使用状态码进行应答。

第(2)步与第(3)步交替进行，直至连接关闭。

5.5.2　FTP操作命令与状态码

无论是FTP的操作命令还是状态码，都是使用21端口发送的，20端口不发送操作命令和状态码，只传递数据。

FTP操作命令分为三类：访问控制命令、传输参数命令和服务命令。其中访问控制命令主要起到发送用户名、密码、更改路径、退出FTP等作用，传输参数命令主要起到设置传输模式、设置传输类型等作用，服务命令主要起到上传、下载、重命名、显示文件、删除等作用。FTP的全部操作命令如表5.11所示。

表5.11　FTP操作命令列表

命令		说明
访问控制命令	USER	标识用户的“用户名”，控制连接建立以后，此命令通常是第一条命令
	PASS	标识用户的“密码”，此命令必须紧随上一条USER命令发送
	ACCT	标识用户的“账号”，某些记账站点在输入密码后仍需要通过账号才能登录
	CWD	更改工作目录
	CWUP	更改工作目录为上一层目录
	SMNT	此命令使用户可以挂载其他文件系统数据结构，而无须重新登录
	REIN	退出登录但不终止当前传输
	QUIT	退出登录并终止当前传输
传输参数命令	PORT	设置传输模式为主动模式，将IP＋端口以点分十进制方式表示，等待服务器连接自己通知的IP端口
	PASV	设置传输模式为被动模式，请求服务器开启监听DTP端口等待自己连接，服务器监听端口将以IP＋端口以点分十进制方式回应
	TYPE	设置传输的类型为二进制类型或ASCII类型
	STRU	设置传输数据的结构为文件(无记录结构)、记录结构、页面结构
	MODE	设置传输模式为流模式、块模式、压缩模式
服务命令	RETR	下载文件到客户端
	STOR	上传文件到服务器
	STOU	上传文件到服务器，并在当前路径中创建唯一名称的结果文件
	APPE	上传文件到服务器，并在服务器创建对应目录
	ALLO	为服务器预留存储空间
	REST	将从指定的文件偏移量处开始重新传输文件
	RNFR	重命名文件的旧文件名
	RNTO	重命名文件的新文件名
	ABOR	终止当前传输
	DELE	删除服务器上的指定文件
	RMD	删除服务器上的指定目录

续表

命令		说明
服务命令	MKD	在服务器上创建目录
	PWD	显示服务器当前工作目录
	LIST	显示服务器当前目录下的文件
	NLST	显示服务器指定目录下的文件
	SITE	服务器提供特定于其系统的功能
	SYST	查询服务器操作系统类型
	STAT	查询当前的设置
	HELP	帮助
	NOOP	空操作,用于服务器答复 OK

当客户端发起了上述操作命令后,服务器将通过状态码进行回应,FTP 的状态码与 HTTP、SMTP 的状态码类似,均是以 XYZ 格式的三位数字表示的,每位数字均有不同含义,且其状态码的设计原则与 SMTP 一致。FTP 的状态码设计原则如表 5.12 所示。

表 5.12 FTP 状态码的设计原则

第一位数字:X 的含义	(1yz)肯定的初步答复
	(2yz)肯定的完成答复
	(3yz)肯定的中间答复
	(4yz)瞬间否定完成答复
	(5yz)永久否定完成答复
第二位数字:Y 的含义	(x0z) 语法错误
	(x1z) 这些是对信息请求的回复,例如帮助或状态
	(x2z) 答复引用控制和数据连接
	(x3z) 身份验证和记账,对登录过程和记账过程的答复
	(x4z) 尚未指定
	(x5z) 文件系统,这些答复指示服务器文件系统相对于请求的传输或其他文件系统操作的状态

由表 5.12 可见,虽然 FTP 与 SMTP 的状态码含义在细节上不完全相同,但设计原则大体一致。FTP 的全部状态码如表 5.13 所示。

表 5.13 状态码列表

状态码	说明
200	命令成功
500	语法错误,命令无法识别
501	参数错误或参数中的语法错误
202	命令在此站点上不可用
502	命令不可用
503	命令顺序错误
504	参数不可用
120	在 n 分钟内准备好服务

续表

状　态　码	说　　明
220	服务就绪
221	服务关闭控制连接
421	服务不可用，正在关闭控制连接
125	数据连接已经打开，传输开始
225	数据连接已经打开，没有进行中的传输
425	无法打开数据连接
226	关闭数据连接
426	连接关闭，传输终止
227	进入被动模式
230	用户登录，继续
530	用户未登录
331	用户名正确，需要密码
332	需要登录账户
532	需要账户来存储文件
110	重新启动标记回复
211	系统状态回复或系统帮助回复
212	目录状态回复
213	文件状态回复
214	帮助消息回复
215	系统消息回复
150	文件状态正常，即将打开数据连接
250	请求的文件操作正常，已完成
257	目录已创建
350	请求的文件操作有待进一步的信息
450	未成功执行请求文件操作，例如文件占用中
550	未成功执行命令，例如找不到文件
451	请求的操作终止。处理中的本地错误
551	请求的操作终止。页面类型未知
452	未成功执行命令，例如系统存储空间不足
552	请求的文件操作终止，例如超出当前目录的存储分配
553	未成功执行命令，例如不允许使用的文件名

5.5.3　FTP的主动模式与被动模式

了解了FTP的操作命令、状态码、双通道工作机制以后，先不要着急去查看FTP的工作流程，由于FTP是双通道的，因此在建立第二个通道的时候别有一番讲究：是让服务器主动发SYN建立数据连接，还是让服务器被动等待SYN连接？

这类似于生活中年轻男女谈恋爱的主动与被动问题，生活中的恋爱一般来说男生都是主动“发起连接”的，当然少数情况下男生是被动“等待连接”的。FTP服务器也是这样

的,有主动和被动之分。假设服务器性别为“男”,客户端性别为“女”,那么服务器(男)应选择主动模式还是被动模式?这取决于客户端(女)的态度。这类似于生活中的女生对男生讲的“你主动点”,或“你等着吧”。这便是 FTP 数据连接主动模式与被动模式的区别,请记住:主动或被动是服务器的行为,但要求使用主动或被动是客户端的行为。接下来看一下主动模式与被动模式的示例。

主动模式的示例如下:

(1) 客户端发起一个 PORT 命令,该命令内携带 6 个数字,使用逗号分隔,这 6 个数字用于声明客户端希望在数据连接所使用的 IP 和端口。

(2) 服务器收到客户端发来的 PORT 命令,计算 6 个数字,得出客户端声明的 IP 与端口,主动使用自己的 20 端口,连接客户端声明的端口。

(3) 三次握手成功之后,服务器回应状态码 200 声明连接建立成功。

主动模式的工作原理如图 5.69 所示。

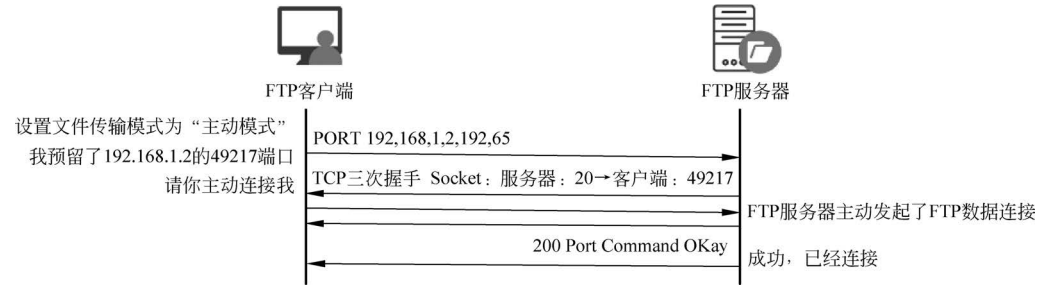


图 5.69 FTP 主动模式的工作原理

在图 5.69 中,客户端发送了 PORT 命令,声明了 6 个数字:192,168,1,2,192,65。其中 192,168,1,2 前 4 个数字固定表示 IP 地址。后面的 192,65 表示端口,经过计算,能够得出端口号为 49217,计算方式为 192×256+65。

服务器得到端口号之后,使用自己的 20 端口主动与客户端的 49217 端口建立 TCP 连接,连接建立完成之后,发送状态码为 200 的消息,表示 PORT 模式的数据连接建立成功。接下来,数据连接将用于发送一次数据(或打印目录,或传送文件)。

被动模式的示例如下:

(1) 客户端发起一个 PASV 命令,声明使用被动模式。

(2) 服务器以 227 状态码返回消息,该消息携带 6 个数字,分别是 192、168、1、1、192、65,使用逗号分隔,这 6 个数字用于声明服务器希望在数据连接所使用的 IP 和端口。

(3) 客户端收到 227 消息之后,计算 6 个数字,得出服务器声明的 IP 与端口,主动使用自己的随机端口连接服务器声明的端口。

(4) 三次握手成功之后,PASV 结束,此时可以直接使用数据连接,无须声明连接建立成功。

被动模式的工作原理如图 5.70 所示。

在图 5.70 中,客户端发送了 PASV 命令,服务器返回了 227 消息,在消息中携带了 6 个数字:192,168,1,1,192,65,这 6 个数字表示服务器预留给客户端建立数据连接的 IP

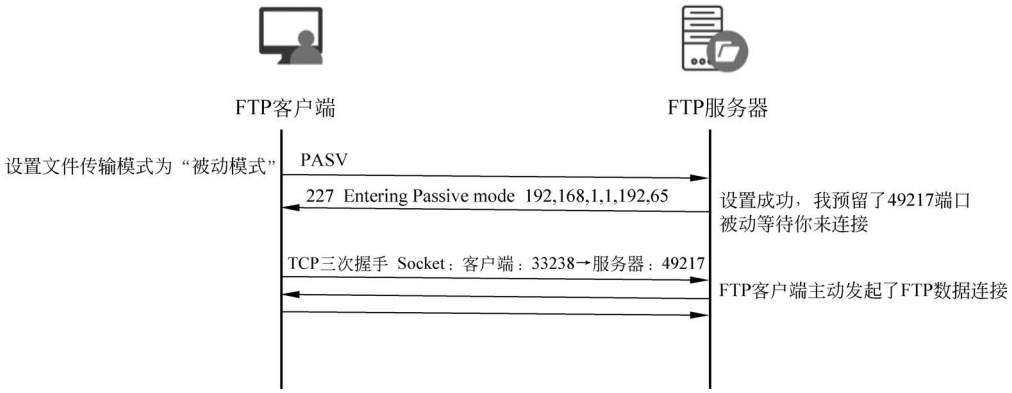


图 5.70　FTP 被动模式的工作原理

地址为 192.168.1.1，端口为 192×256＋65＝49217，此时服务器已经预留了 192.168.1.1：49217 被动等待客户端发起连接。

客户端得到端口号之后，使用自己的随机端口 33238 主动与服务器的 49217 端口建立 TCP 连接，连接建立完成之后，PASV 模式的数据连接建立成功。接下来，数据连接将用于发送一次数据（或打印目录，或传送文件）。

【经验分享】 *被动模式没有使用到 20 端口，FTP 只有主动模式使用 20 端口。*

为什么需要用 192×256＋65＝49217 这样的计算方式？为什么 192 需要乘以 256，不需要乘以 65？49217 是将 192 与 65 这两个数字的二进制拼接组合在一起的结果。192 表示二进制数 11000000，65 表示二进制数 01000001，拼接在一起的结果为 1100000001000001。将拼接之后的数字转换为十进制，即为 49217，由于拼接之后，192 的二进制被放在了前面，后面添加了 8 位二进制数字，相当于被扩大了 2^8 倍，即 256 倍。因此，需要将 192 乘以 256，而不用乘以 65。

5.5.4　FTP 工作流程详解

FTP 进行文件传输的基本工作流程主要分为 4 个阶段：连接建立、身份认证、命令交互和连接断开，如图 5.71 所示。

1. 连接建立阶段

该阶段 FTP 客户端通过 TCP 三次握手与 FTP 服务器端建立连接。

2. 身份认证阶段

三次握手成功以后，FTP 服务器需要客户端进行身份认证，向客户端发送身份认证请求。客户端需要向 FTP 服务提供登录所需的用户名和密码。FTP 服务器对客户端输入的用户名和密码都会给出相应的应答。如果客户端输入的用户名和密码正确，将成功登录 FTP 服务器，此时进入 FTP 会话，信息如图 5.72 所示。

在三次握手连接建立成功以后，首先服务器会发送一个 220 状态码表示连接建立成功。随后客户端会通过 USER 命令发送用户名，服务器会返回 331 表示用户名正确，继续请求密码，此时客户端通过 PASS 命令发送密码，如果密码正确，服务器将返回 230 状

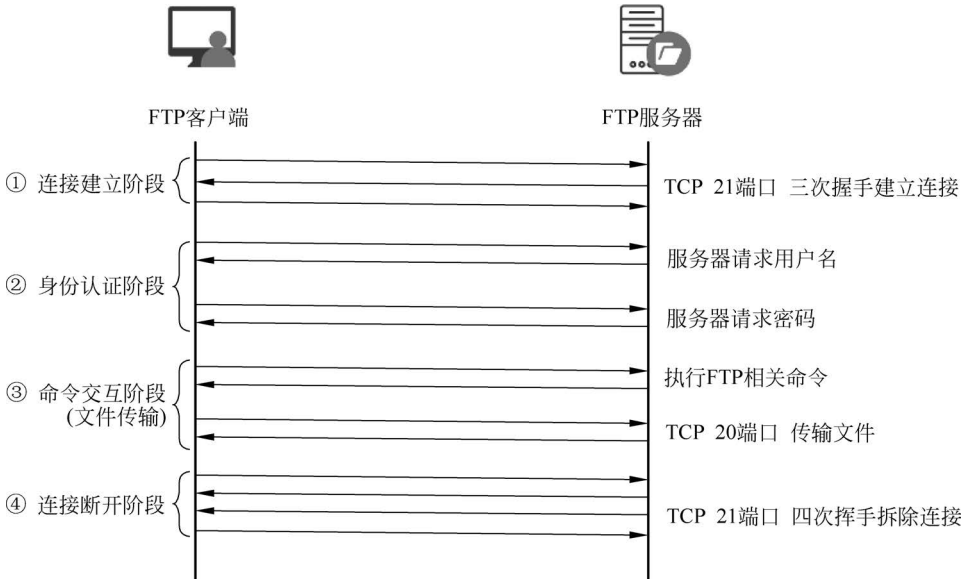


图 5.71 FTP 的工作原理

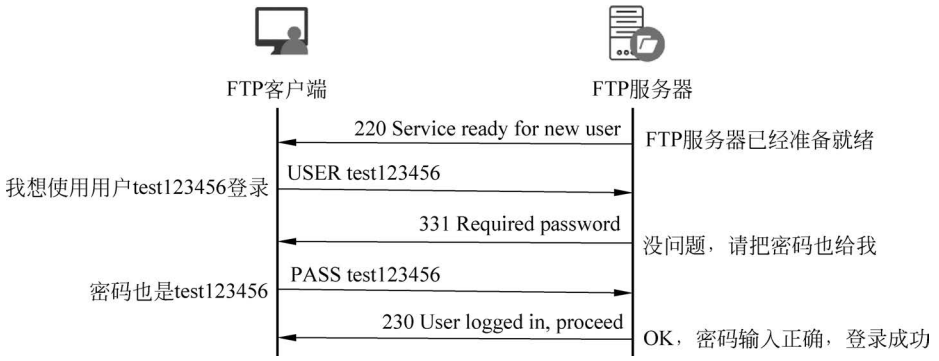


图 5.72 FTP 认证示意图

态码表示登录成功。在这个阶段可以发现一些安全隐患：用户名和密码的传输是明文的，因此某些安全设备能够对内网的 FTP 弱口令进行发现与识别，都是基于流量去识别的，信息如图 5.73 所示。

3. 命令交互阶段

在 FTP 会话中，用户可以执行 FTP 命令进行文件传输，如查看目录信息、上传或下载文件等。客户端输入要执行的 FTP 命令后，服务器同样会给出应答。如果输入的命令正确，服务器会将命令的执行结果返回给客户端。执行结果返回完成后，服务器继续给出应答。在这个阶段会使用到 TCP 20 端口用于建立 FTP 数据连接，并传输数据。

在登录成功以后，客户端可以通过 SYST 询问 FTP 服务器运行在什么样的操作系统上，服务器返回 215 状态码，并附状态消息表示这是一个基于 UNIX 操作系统的 FTP 服务器。客户端通过 OPTS 声明传输使用的字符编码方式为 UTF-8，用于防止目录与文件

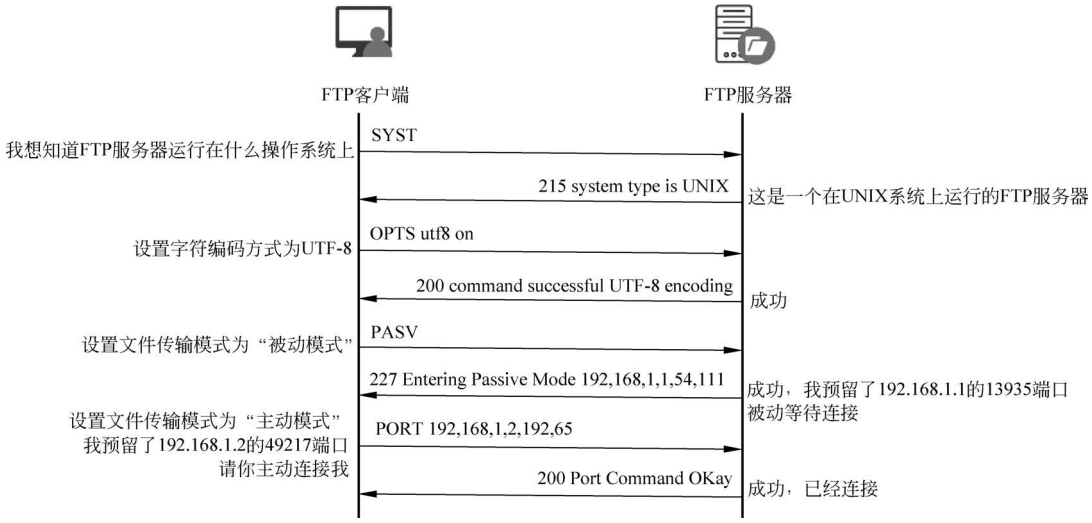


图 5.73　FTP 认证成功示意图

名显示乱码，服务器通过 200 表示设置字符编码方式成功。

命令交互阶段除常规命令交互外，还需一个重要的命令交互步骤——建立数据连接。图 5.74 和图 5.75 分别展示了主动模式和被动模式的交互流程。在实际使用中，交互模式基本都是固定的，由于一般部署防火墙禁止从外向内访问流量，FTP 部署通常使用主动模式。

【注意】 无论使用的数据连接是通过主动还是被动模式建立的，数据连接都仅被使用一次，用完就关闭，类似于生活中的一次性餐具，用完就丢弃。一个数据连接可以用于打印一次目录或传送一次文件。图 5.74 和图 5.75 分别展示了主动和被动模式数据连接建立的过程。

图 5.74 展示了通过主动模式建立数据连接后，客户端通过 21 端口发送了 LIST 命令，希望查看当前访问的 FTP 服务器上的文件和目录信息，服务器通过 21 端口，使用 150 状态码返回表示请求被正确处理，同时通过 20 端口将要发送的数据发送给客户端，而后进行 4 次断开，数据连接被消耗（每个数据连接只能被使用一次）；同时服务器会再次通过 21 端口发送 226 状态码表示数据已经发送完毕。如果后续客户端再次执行上传或下载命令，需要再次通过 PORT 命令建立一个新的数据连接。

图 5.75 展示了通过被动模式建立数据连接后的交互，与主动模式类似，发送一次数据内容以后，即断开数据连接，等待重新建立新的，这里不再赘述。

4. 连接断开阶段

当客户端不再与 FTP 服务器进行文件传输时，需要断开连接。客户端向 FTP 服务器发送断开连接请求，服务器收到断开连接请求后给出相应的应答。

总结如下。

（1）FTP 使用两个连接：21 端口控制连接与 20 端口数据连接。

（2）FTP 客户端发送操作命令，FTP 服务器返回状态码。

图 5.74 FTP 主动模式建立连接示意图

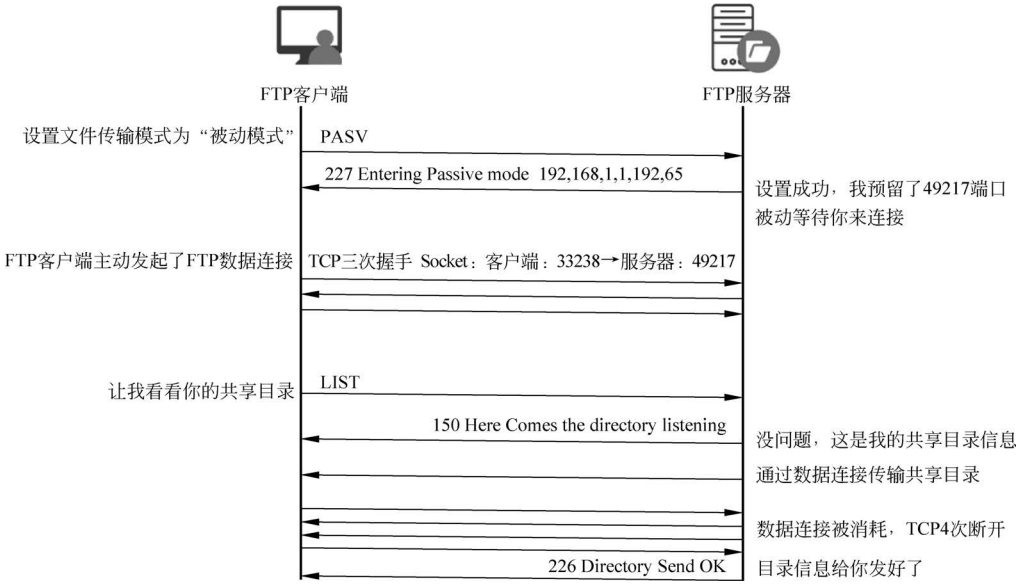


图 5.75 FTP 被动模式建立连接示意图

(3) FTP 共有 4 个阶段,分别是连接建立、身份认证、命令交互(文件传输)、连接断开。

(4) FTP 每一个数据连接只能使用一次,使用完毕后即被消耗,消耗后重新建立一个数据连接。

(5) FTP 在文件传输阶段有主动模式与被动模式之分。

5.5.5 FTP 分析方法

科来 CSNAS 提供两种分析邮件数据包的方式,分别是日志分析和数据流解码。其

中日志分析方式适合快速定位某个 TCP 会话数据流，数据流解码方式适合用户详细观察 FTP 数据流的交互内容。

1. 日志分析

当 FTP 流量过多，不方便依次查看 TCP 会话数据流进行分析时，可以在“日志”视图选中左侧的“FTP 传输”观察 FTP 日志，这些日志全部是基于捕获到的数据包生成的，并以列的形式展示，每一列为一次 FTP 数据连接的交互，列中包括双方通信的地址、端口、账号、操作类型、文件名、字节数等信息，用户可以使用日志功能快速浏览网络中出现的 FTP 文件传输行为。当日志数量过多时，日志支持以关键字作为条件进行搜索，快速列出某些特定的 FTP 传输（如文件名或登录账号），如图 5.76 所示。

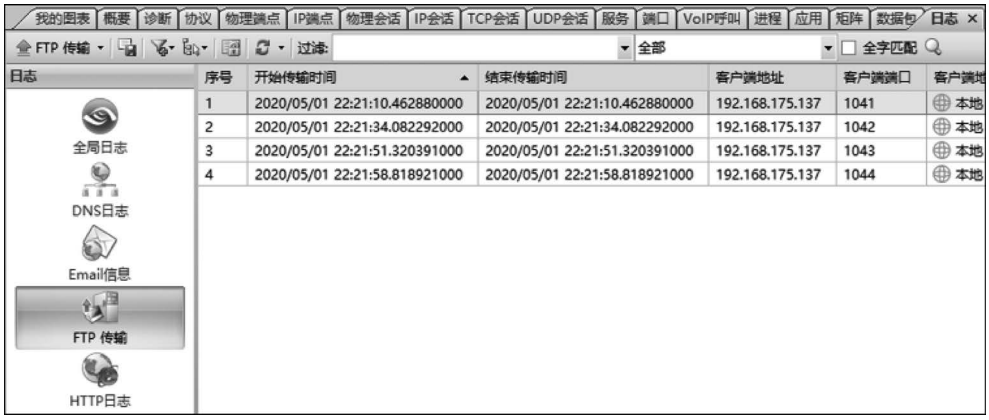


图 5.76　通过日志对 FTP 流量进行分析

右击列头，选择“更多”，可以对 FTP 日志中的列进行选择，启用/关闭某些列，如图 5.77 所示。

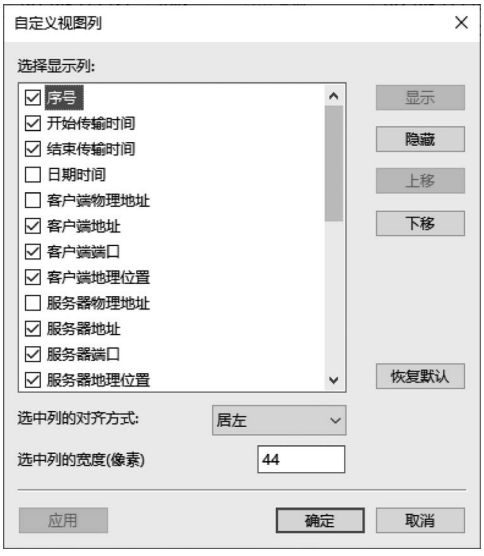


图 5.77　自定义 FTP 日志中的显示列

当发现某条可疑的 FTP 日志时,双击这条日志,即可弹出新窗口,该窗口显示这条日志对应的 TCP 会话数据流,如图 5.78 所示。

图 5.78 使用 TCP 数据流分析 FTP 流量

2. 数据流解码

在“TCP 会话”视图观察 FTP 的 TCP 控制连接会话,如网络中出现的会话过多,可以在过滤框中输入:“session.port=21”语句过滤所有 21 端口的 TCP 会话,如图 5.79 所示。

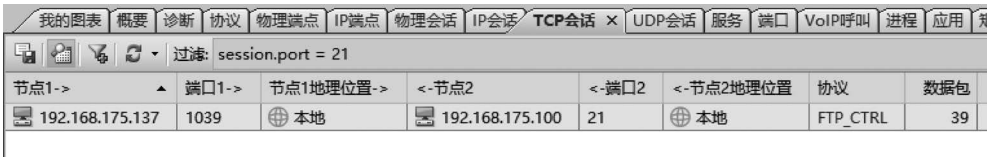


图 5.79 在“TCP 会话”视图过滤 21 端口的会话

找到需要分析的 TCP 会话以后,单击下方的“数据流”直接观察 TCP 通信数据流,在这里按照前面章节叙述的工作流程观察 FTP 流量,该方式能够快速分析这条 FTP 会话的交互内容,更适合对 FTP 流量分析的初学者,如图 5.80 所示。

图 5.80 中的前两行由软件自动生成,声明了通信节点双方所使用的 IP 地址和端口号,并使用颜色区分双方发送的数据。其中深色字体为节点 1 发送数据,浅色字体为节点 2 发送数据,十分直观。通过端口号不难判断,使用 1039 端口的节点 1 设备为客户端,使用 21 端口的节点 2 设备为服务器。

首先服务器发送 220 表示服务就绪,然后客户端通过 anonymous 账号进行登录(FTP 匿名账号,无须密码),客户端执行 PORT 命令指定通信的端口为 $4\times256+17=1041$,建立数据连接,服务器主动使用 20 端口建立这个连接,如图 5.81 所示。

连接建立以后,客户端通过 RETR 命令下载 download1.txt,服务器连续返回状态码 150 和 226 分别表示“开始以 ASCII 模式传输 download1.txt”和“传输完毕”。可想而知,在这两个指令之间,FTP 通过数据连接将 download1.txt 文件发送给了客户端。随即这个数据连接被废弃,后续客户端又执行了几次 PORT 命令建立了几个数据连接,分别执行了一些上传和下载操作,后续的上传与下载操作与之前的类似,这里就不再赘述了。图 5.82 展示了数据连接 1041 端口所传输的文件内容。

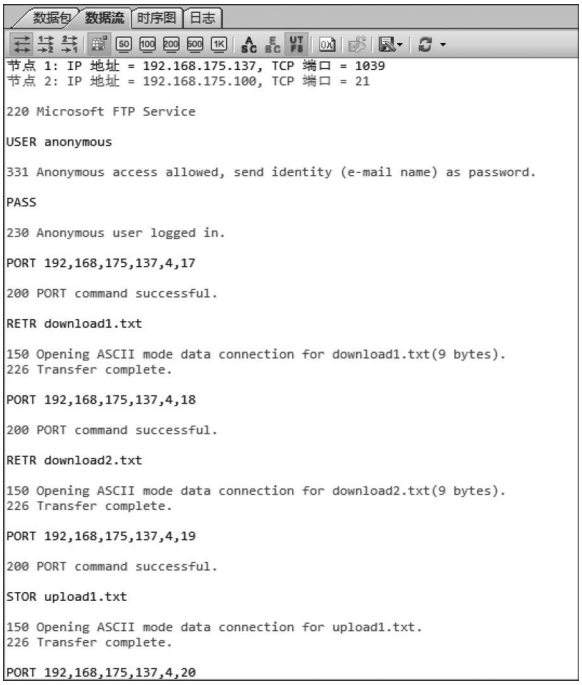


图 5.80　使用 TCP 数据流功能分析 FTP 控制连接会话

图 5.81　通过 1041 端口建立的 FTP 数据连接

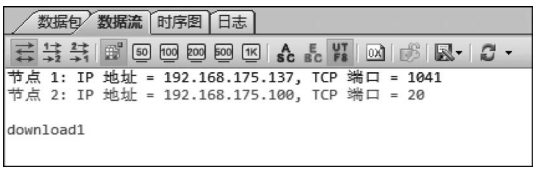


图 5.82　通过 1041 端口传输的文件内容

通过分析可知，这个连接中传输的为 download1.txt 的全部内容，文件中的内容为一串字符 download1，到这里已经分析到了本次 FTP 传输文件的内容。

在实际分析工作中，可能有使用 FTP 传输一些非法内容的情况（包括音频、视频、文档等），传输这类文件时，无法直接通过“数据流”视图观察到文件的内容，需要将文件进行进一步的处理，下一小节将讨论如何对 FTP 传输的内容进行分析。

5.5.6 通过 FTP 流量还原 FTP 交互的文件

当遇到 FTP 传输的无法通过肉眼观察的恶意程序(如后门程序)时,通过科来 CSNAS 可以将恶意程序的样本完全还原到计算机本地。假设 5.5.5 节讨论的 download1.txt 为恶意文件,接下来展示如何将恶意文件样本还原到本地。

(1) 在流量中找到对应会话(FTP 数据连接会话),打开 TCP 数据流分析界面。

通过对 21 端口的分析,计算某个文件的对应传输端口,寻找对应端口的 TCP 会话,双击打开会话分析窗口,或在 FTP 日志界面双击某条日志,打开会话分析的"数据流"视图,如图 5.83 所示。

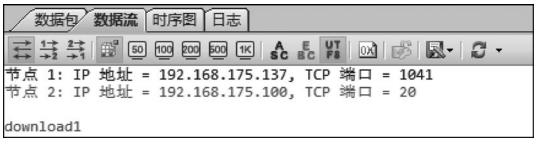


图 5.83 FTP 数据连接会话内容

(2) 在"数据流"视图中,单击 OX 按钮,启动 HEX 分析界面,如图 5.84 所示。

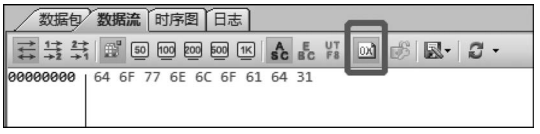


图 5.84 启用 HEX 分析

(3) 按快捷键 Ctrl+A,将右侧的 HEX 全部选中,如图 5.85 所示。

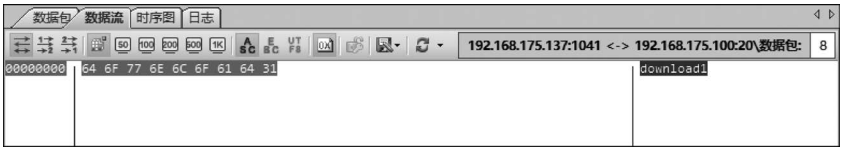


图 5.85 全部选中右侧的 HEX 内容

图 5.86 科来 CSNAS 数据流解码导出操作

(4) 右击,在打开的快捷菜单中选择"导出选择"选项,如图 5.86 所示。

(5) 对文件按照原文件名进行保存,注意选择保存类型为 All Files,如图 5.87 所示。

(6) 文件保存后,对其进行分析,单击"打开文件"按钮,如图 5.88 和图 5.89 所示。

根据以上步骤,可以完全将 FTP 传输中的内容还原。

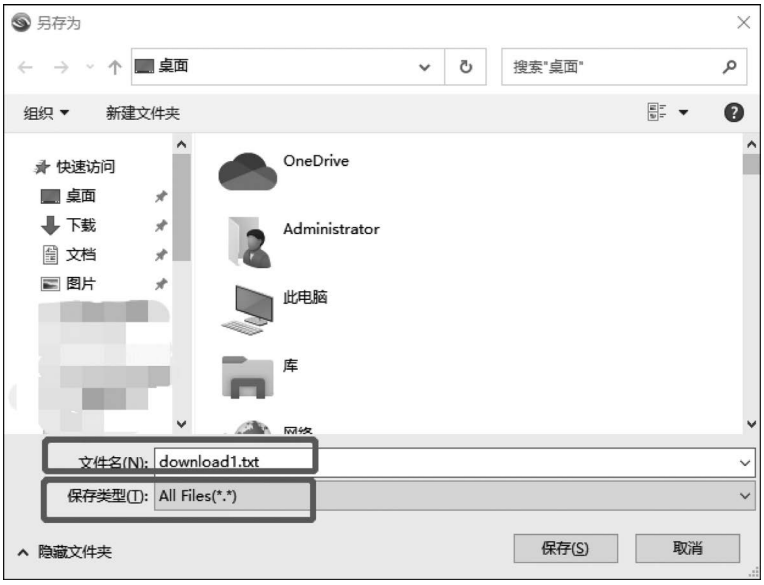


图 5.87　将 HEX 存储为 download1.txt

图 5.88　保存成功提示

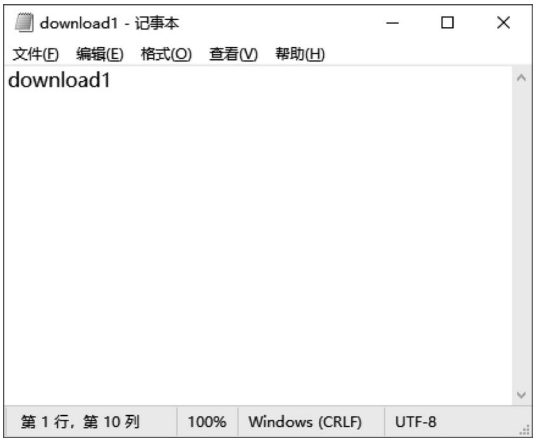


图 5.89　打开还原后的文件

5.6 实验：HTTP/HTTPS 流量分析

使用科来 CSNAS 启动实时捕获数据包，访问一个非 HTTPS 的网站，如 http://www.colasoft.com.cn，然后停止捕获。

在“TCP 会话”分页中，通过如下过滤语句搜索刚刚访问的 HTTP 网站(如果抓包时访问的是其他 HTTP 网站，则需将过滤语句中的 colasoft.com.cn 替换为其他网站名称)：

```
session.ep2.find(/colasoft.com.cn/)
```

过滤后，结果如图 5.90 所示。如果无法搜索到结果，请检查“IP 地址显示格式”是否为“显示 IP 名字和地址”。

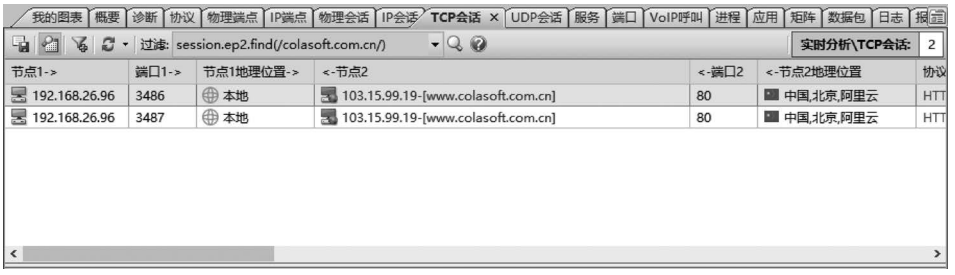


图 5.90 过滤结果

找到相关会话后，单击列表中选中的会话，在弹出的子视图中选择“数据流”，如图 5.91 所示。

图 5.91 “数据流”视图

图 5.91 为分析 HTTP 流量的常用界面，界面内显示的为 HTTP 会话交互的内容，第 2 自然段为节点 1 发送的数据，第 3 自然段为节点 2 发送的数据，使用快捷键 Ctrl+鼠标滚轮，可以调整字体大小。图 5.91 中的会话内容可以大致解读为：客户端向服务器 www.colasoft.com.cn 的根目录发起了 GET 请求，服务器响应了 200 OK 状态码，交互

协议为 HTTP 1.1。

在图 5.91 中的工具栏可以选择编码的方式，软件支持以 ASCII、EBCDIC、UTF-8 进行编码，若在科来 CSNAS 中选择的编码方式与网站的编码方式一致，则可以正常显示网站中传输的中文字符，如图 5.92 所示。

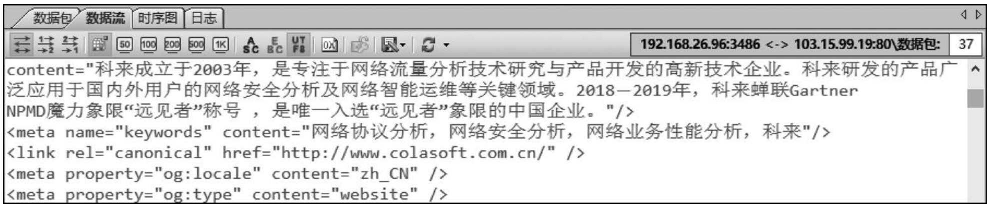


图 5.92　数据流编码转换界面

单击工具栏的 0X 按钮，可以进入十六进制视图，如图 5.93 所示。可以查询会话中传输的原始十六进制信息，双方发送的数据仍以颜色进行区分。

利用十六进制视图，可以通过选中原始十六进制信息进行导出的方式，对 HTTP 内传输的文件进行恢复，例如图片、音乐、视频、可执行程序、SSL 证书等。

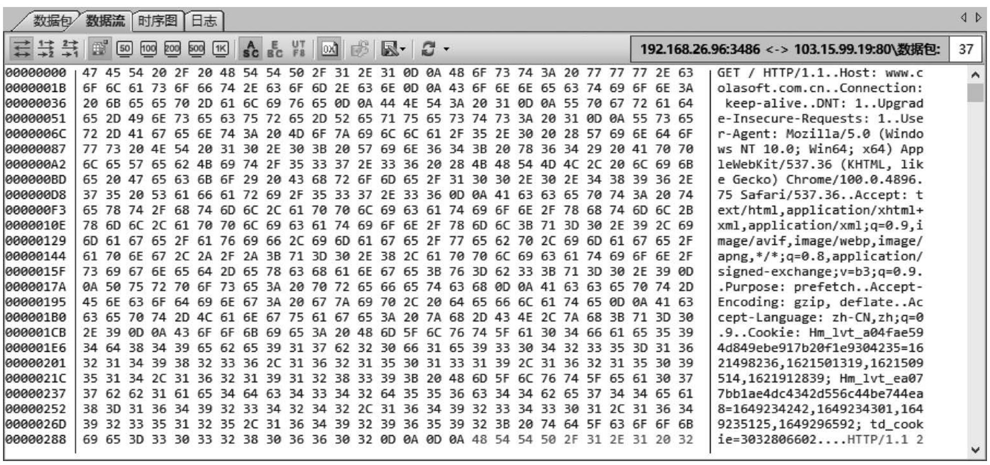


图 5.93　数据流十六进制转换界面

当网络中存在大量的 HTTP 会话时，逐条分析显然不是方便快捷的方法。科来 CSNAS 提供了"HTTP 日志"功能，可供用户快速浏览网络流量中出现的 HTTP 会话。切换至"日志"视图，在该视图左侧选择"HTTP 日志"选项，如图 5.94 所示：

图 5.94 中展示的日志每行为一次 HTTP 交互。该日志并非操作系统日志或 HTTP 服务器日志，而是基于网络流量，通过对网络流量进行统计生成的日志。日志内包含时间、地址、端口、请求方法、请求 URL、响应状态码、User-Agent 等常见的 HTTP 流量需要关注的内容。用户可以通过右击表头对统计显示的列进行调整，也可以使用 Ctrl＋Shift＋A 快捷键对列宽进行自动调整。

日志功能支持字符搜索，若要搜索所有 POST 请求，则可在"过滤"框中输入 POST，然后按回车键进行过滤，过滤结果如图 5.95 所示。

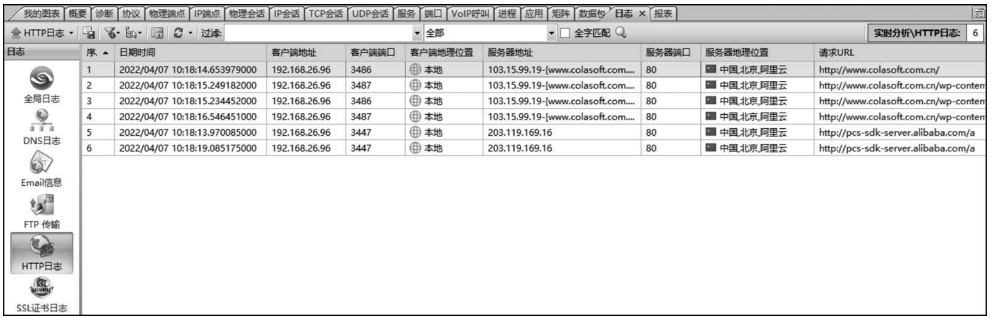


图 5.94 HTTP 日志分析界面

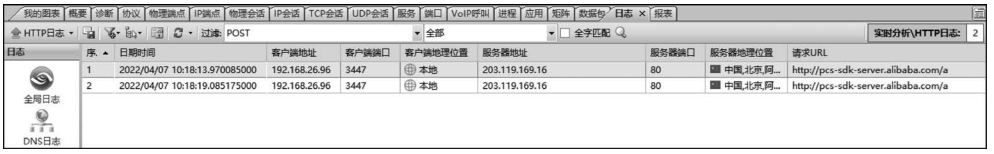


图 5.95 过滤 POST 请求

使用搜索过滤可以快速找到用户关心的会话,单击日志中的 URL 可以直接跳转到该 URL,因此在进行恶意 URL 流量分析时,请避免误触单击恶意 URL。当需要对某条会话进行进一步分析时,可以双击会话,调出针对该会话的分析窗口,对该会话的数据流、数据包进行进一步分析。双击后的分析窗口如图 5.96 所示。

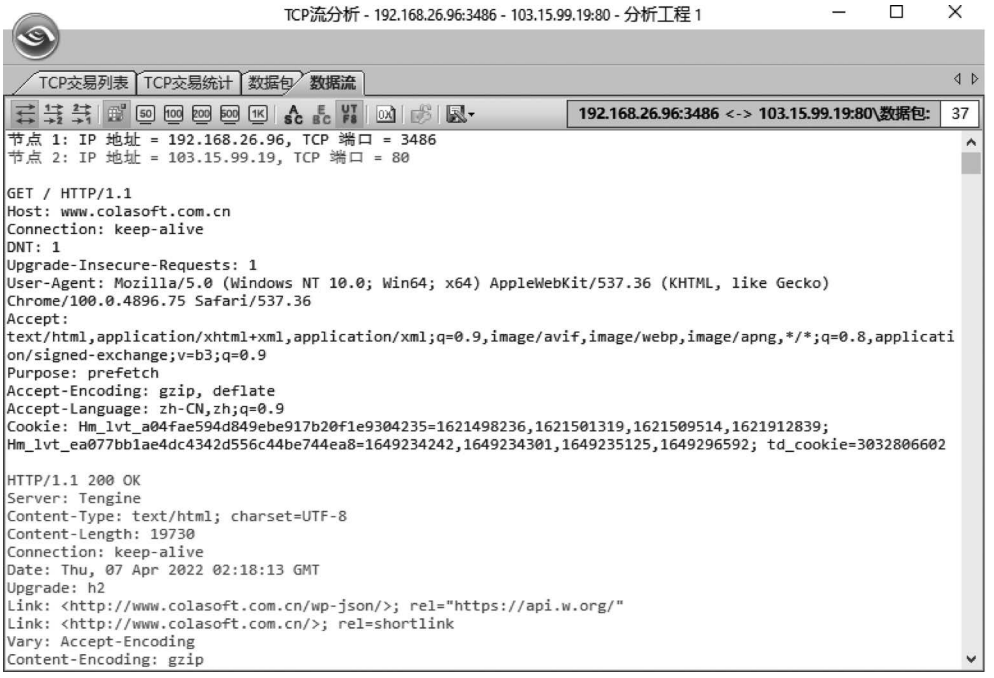


图 5.96 HTTP 日志详细分析界面

以上内容为HTTP流量的常见分析方法，要求读者掌握对TCP会话数据流以及HTTP日志进行分析的能力，掌握对HTTP请求中的常见参数、HTTP应答中的状态码和实体内容的解读能力。

5.7　习　　题

1. 常见的HTTP请求方法与状态码有哪些？分别表示什么含义？
2. HTTP如何维持长连接？
3. 如何分辨非标准HTTP流量？
4. 简述两种HTTP流量分析方法。
5. 如何获取FTP与SMTP/POP3/IMAP的登录用户名与密码？