MIME嗅探，Encoding嗅探，以及UTF-7的简单介绍

ch1ps79 / 2019-03-07 21:54:00 / 浏览数 1238 新手 入门资料 顶(0) 踩(0)

最近开始看hackerone
101的教学视频，希望从这个教学入手来建立和联系自己的知识树，在看完第二节的web介绍后，对web方向的学习有了大致认识，在此推荐稍微有一点英语水平的同学观看
101的教学视频(词汇并非很难，一般水准即可理解，结合搜索引擎）

开头分别介绍了web的一些基础知识，http协议，cookies，html，等一些比较基础的web知识，但其中有几点却让我比较感兴趣，本文将着重讨论一下几点。

1.MIME sniffing（MIME 嗅探）

产生这个问题的主要原因是，http报文中的content-type头在早期版本的浏览器中(例如IE7/6），浏览器并不光会凭靠content-type头进行不同类型的解析，还会对respon
text/plain ，将文件内容直接显示，response中内容为

<script> alert(1); </script> IE浏览器会将其自动嗅探，并认为是text/html类型，并执行相应的渲染逻辑，这就很容易引发XSS攻击。
攻击者可以将图片文件内容写入xss攻击语句，并上传到共享的网站上，当用户请求该文件时，老旧浏览器收到response进行违背content-type的html解析，从而触发MIM
#2.Encoding sniffing （编码嗅探）
产生这一问题的主要原因是UTF-7的存在，以及在没有特别指定格式编码的情况下，浏览器会根据编码格式进行相对应的解码和编码过程。 例如在xss
paylaod中插入UTF-7编码从而达到逃逸过滤的效果。 假如我们的xss payload为<script>alert(1);</script>

但是后台过滤 <符号或者是>符号

我们就可以通过+ADw-script+AD4-alert (1); +ADw-script+AD4-

来通过编码嗅探来绕过过滤。（插一句题外话，其实对于很多的过滤的情况可以尝试各种编码或者反过滤情况来逃脱过滤这一环节）

同时大致介绍一下UTF-7这一编码

UTF-7首次被提出是在一个实验性的通信协议里（RFC 1642，A Mail-Safe Transformation Format of Unicode），这份RFC（Request for
Comments）提案后来因RFC 2152的提出而被取代（RFC 2152本身为新闻型（informational）的文案）。在RFC
2152当中明确的指出该份RFC本身不为互联网的标准做出任何明确的定义（明列于文案前头的Status of this Memo）。尽管这份RFC 2152在IANA（Internet
Assigned Numbers Authority）的字符集列表里被引述为UTF-7，然而UTF-7本身并非Unicode的标准之一，即使在当前最新的Unicode
5.0里也仅列出UTF-8、UTF-16和UTF-32。

如同引言所提到的，由于在过去SMTP的传输仅能接受7比特的字符，而当时Unicode并无法直接满足既有的SMTP传输限制，在这样地背景下UTF-7被提出。严格来说UTF-

有些字符本身可以直接以单一的ASCII字符来呈现。第一个组群被称作"direct characters"，其中包含了62个数字与英文字母，以及包含了九个符号字符：' ( ) , - . /
: ?。这些"direct characters"被认为可以很安全的直接在文件里呈现。另一个主要的组群称作"optional direct
characters"，其中包含了所有可被打印的字符，这些字符在U+0020～U+007E之间，除了~ \ +和空白字符以外。这些"optional direct
characters"的使用虽可减少空间的使用也可增加人的可阅读性，但却会因为一些不良设计的邮件网关而会产生一些错误，导致必须使用额外的转义字符。

空白字符、Tab字符、以及换行字符一般虽也可直接是为单一的ASCII字符来使用，然而，若是邮件中有使用了编码过的字符串，则必须特别注意这些字符有无被使用在其他

其他的字符则必须被编码成UTF-16然后转换为修改的Base64。这些区块的开头会以+符号来标示，结尾则以任何不在Base64里定义的字符来标示。若是在Base64区块之后

上面这么长一段话呢，大概就是UTF-7的来历和编码规则和表达方式，简单来说的话就是UTF-8是将一个无论什么样的字符都统一存放到8位的字节单位里面，而UTF-7则不

第一类是直接字符，包含了62个数字与英文字符（0-9，a-z，A-Z）以及
' ( ) , - . / : ?。

再将其他的字符分成了选择项直接字符

直接字符直接用ASCII码表示，其他的选项直接字符则编码成UTF-16后再转化成Base64，并且使用了这一字符的区块以+为开始，以任何不在Base64定义的字符为结束。

上一篇：CVE-2018-12794类型混... 下一篇：一个JS沙箱逃逸漏洞
1. 0 条回复
   - 动动手指，沙发就是你的了！

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板