Atitit [HTTP](http://blog.csdn.net/wwwsq/article/details/7255062) 认证机制基本验证 (Basic Authentication) 和摘要验证 (Digest Authentication)attilax总结

[1.1. 最广泛使用的是基本验证 (Basic Authentication) 和摘要验证 (Digest Authentication)。 1](#_Toc21579)

[1.2. 关于HTTP AUTH的文档不多。 1](#_Toc10074)

[1.3. 什么是HTTP基本认证 1](#_Toc32604)

[1.4. 适用场合 路由器 摄像头 2](#_Toc8632)

[1.5. 其他认证 　　除了基本认证(Basic Authentication), 还有摘要认证digest authentication, WSSE(WS-Security)认证 4](#_Toc8618)

**[1.6.](#_Toc31285)** **[摘要访问认证](#_Toc31285)** [4](#_Toc31285)

[1.7.](#_Toc32542) **[.服务器响应](#_Toc32542)** [服务端返回401未验证的状态，并且返回WWW-Authenticate信息,包含了验证方式](#_Toc32542)**[Digest，realm，qop，nonce，opaque](#_Toc32542)**[的值。其中： 4](#_Toc32542)

[1.8.](#_Toc15798) **[3.客户端请求  (用户名 "Mufasa", 密码 "Circle Of Life")](#_Toc15798)** [5](#_Toc15798)

## 最广泛使用的是基本验证 (Basic Authentication) 和摘要验证 (Digest Authentication)。

## 关于HTTP AUTH的文档不多。

RFC在 http://www.ietf.org/rfc/rfc2617.txt

wiki在 http://en.wikipedia.org/wiki/Basic\_access\_authentication

## 什么是HTTP基本认证

　　桌面应用程序也通过HTTP协议跟Web服务器交互， 桌面应用程序一般不会使用cookie, 而是把 "用户名+冒号+密码"用BASE64算法加密后的字符串放在http request 中的header Authorization中发送给服务端， 这种方式叫HTTP基本认证(Basic Authentication)

　　当浏览器访问使用基本认证的网站的时候， 浏览器会提示你输入用户名和密码，如下图

使用HTTP AUTH需要在server端配置http auth信息（一般是webserver启动的时候从配置文件里面读取相关信息）。我用中文简述一下http auth的过程：

* 客户端发送http请求
* 服务器发现配置了http auth，于是检查request里面有没有"Authorization"的http header
* 如果有，则判断Authorization里面的内容是否在用户列表里面，Authorization header的典型数据为"Authorization: Basic jdhaHY0="，其中Basic表示基础认证， jdhaHY0=是base64编码的"user:passwd"字符串。
* 如果没有，或者用户密码不对，则返回http code 401页面给客户端
* 标准的http浏览器在收到401页面之后，应该弹出一个对话框让用户输入帐号密码；并在用户点确认的时候再次发出请求，这次请求里面将带上Authorization header

一次典型的访问场景是：

* 浏览器发送http请求（没有Authorization header）
* 服务器端返回401页面
* 浏览器弹出认证对话框
* 用户输入帐号密码，并点确认
* 浏览器再次发出http请求（带着Authorization header）
* 服务器端认证通过，并返回页面
* 浏览器显示页面

## 适用场合 路由器 摄像头

使用http auth的场景不会用cookie，也就是说每次都会送帐号密码信息过去。然后我们都知道base64编码基本上等于明文。这削弱了安全。

由于种种缺点，http auth现在用的并不多。不过在路由器等场合还是有应用的，原因是http auth最简单，使用起来几乎是零成本。

**BASIC认证概述**

在HTTP协议进行通信的过程中，HTTP协议定义了基本认证过程以允许HTTP服务器对WEB浏览器进行用户身份证的方法，当一个客户端向HTTP服务 器进行数据请求时，如果客户端未被认证，则HTTP服务器将通过基本认证过程对客户端的用户名及密码进行验证，以决定用户是否合法。客户端在接收到HTTP服务器的身份认证要求后，会提示用户输入用户名及密码，然后将用户名及密码以BASE64加密，加密后的密文将附加于请求信息中， 如当用户名为anjuta，密码为：123456时，客户端将用户名和密码用“：”合并，并将合并后的字符串用BASE64加密为密文，并于每次请求数据 时，将密文附加于请求头（Request Header）中。HTTP服务器在每次收到请求包后，根据协议取得客户端附加的用户信息（BASE64加密的用户名和密码），解开请求包，对用户名及密码进行验证，如果用 户名及密码正确，则根据客户端请求，返回客户端所需要的数据;否则，返回错误代码或重新要求客户端提供用户名及密码。

**二．**   **BASIC认证的过程**

1．  客户端向服务器请求数据，请求的内容可能是一个网页或者是一个其它的MIME类型，此时，假设客户端尚未被验证，则客户端提供如下请求至服务器:

Get /index.html HTTP/1.0  
Host:www.google.com

2．  服务器向客户端发送验证请求代码401,服务器返回的数据大抵如下：

HTTP/1.0 401 Unauthorised  
Server: SokEvo/1.0  
WWW-Authenticate: Basic realm="google.com"  
Content-Type: text/html  
Content-Length: xxx

3．  当符合http1.0或1.1规范的客户端（如IE，FIREFOX）收到401返回值时，将自动弹出一个登录窗口，要求用户输入用户名和密码。

4．  用户输入用户名和密码后，将用户名及密码以BASE64加密方式加密，并将密文放入前一条请求信息中，则客户端发送的第一条请求信息则变成如下内容：

Get /index.html HTTP/1.0  
Host:www.google.com  
Authorization: Basic xxxxxxxxxxxxxxxxxxxxxxxxxxxx  
  
注：xxxx....表示加密后的用户名及密码。

5．  服务器收到上述请求信息后，将Authorization字段后的用户信息取出、解密，将解密后的用户名及密码与用户数据库进行比较验证，如用户名及密码正确，服务器则根据请求，将所请求资源发送给客户端：

**三．**         **BASIC认证的缺点**

HTTP基本认证的目标是提供简单的用户验证功能，其认证过程简单明了，适合于对安全性要求不高的系统或设备中，如大家所用路由器的配置页面的认证，几乎 都采取了这种方式。其缺点是没有灵活可靠的认证策略，如无法提供域（domain或realm）认证功能，另外，BASE64的加密强度非常低，可以说仅 能防止sohu的搜索把它搜到了。当然，HTTP基本认证系统也可以与SSL或者Kerberos结合，实现安全性能较高（相对）的认证系统

## 其他认证 　　除了基本认证(Basic Authentication), 还有摘要认证digest authentication, WSSE(WS-Security)认证

## ****摘要访问认证****

是一种协议规定的Web服务器用来同网页浏览器进行认证信息协商的方法。它在密码发出前，先对其应用哈希函数，这相对于HTTP基本认证发送明文而言，更安全。从技术上讲，摘要认证是使用随机数来阻止进行密码分析的MD5加密哈希函数应用。它使用HTTP协议。

## ****.服务器响应**** 服务端返回401未验证的状态，并且返回WWW-Authenticate信息,包含了验证方式****Digest，realm，qop，nonce，opaque****的值。其中：

****Digest****：认证方式；

****realm****：领域，领域参数是强制的，在所有的盘问中都必须有，它的目的是鉴别SIP消息中的机密，在SIP实际应用中，它通常设置为SIP代理服务器所负责的域名；

****qop****：保护的质量，这个参数规定服务器支持哪种保护方案，客户端可以从列表中选择一个。值 “auth”表示只进行身份查验， “auth-int”表示进行查验外，还有一些完整性保护。需要看更详细的描述，请参阅RFC2617；

****nonce****：为一串随机值，在下面的请求中会一直使用到，当过了存活期后服务端将刷新生成一个新的nonce值；

****opaque****：一个不透明的（不让外人知道其意义）数据字符串，在盘问中发送给用户。

## ****3.客户端请求  (用户名 "Mufasa", 密码 "Circle Of Life")****

客户端接受到请求返回后，进行HASH运算，返回Authorization参数

其中：****realm，nonce，qop****由服务器产生；

****uri****：客户端想要访问的URI；

****nc****：“现时”计数器，这是一个16进制的数值，即客户端发送出请求的数量（包括当前这个请求），这些请求都使用了当前请求中这个“现时”值。例如，对一个给定的“现时”值，在响应的第一个请求中，客户端将发送“nc=00000001”。这个指示值的目的，是让服务器保持这个计数器的一个副本，以便检测重复的请求。如果这个相同的值看到了两次，则这个请求是重复的；

****cnonce****：这是一个不透明的字符串值，由客户端提供，并且客户端和服务器都会使用，以避免用明文文本。这使得双方都可以查验对方的身份，并对消息的完整性提供一些保护；

****response****：这是由用户代理软件计算出的一个字符串，以证明用户知道口令

HTTP AUTH 那些事 - 王绍全的博客 - 博客频道 - CSDN.NET.html

WebApi接口安全认证——HTTP之摘要认证 - 尽善而为 - ITeye技术网站.html

详解HTTP中的摘要认证机制 - tenfyguo的技术专栏 - 博客频道 - CSDN.NET.html