第13章

摘要认证

基本认证便捷灵活，但极不安全。用户名和密码都是以明文形式传送的，[1]也没有采取任何措施防止对报文的篡改。安全使用基本认证的唯一方式就是将其与 SSL 配合使用。

摘要认证与基本认证兼容，但却更为安全。本章主要介绍摘要认证的原理和实际应用。尽管摘要认证还没有得到广泛应用，但对实现安全事务来说，这些概念是非常重要的。

13.1 摘要认证的改进

摘要认证是另一种 HTTP 认证协议，它试图修复基本认证协议的严重缺陷。具体来说，摘要认证进行了如下改进。

- 永远不会以明文方式在网络上发送密码。
- 可以防止恶意用户捕获并重放认证的握手过程。
- 可以有选择地防止对报文内容的篡改。
- 防范其他几种常见的攻击方式。

摘要认证并不是最安全的协议。[2]摘要认证并不能满足安全 HTTP 事务的很多需求。对这些需求来说，使用传输层安全（Transport Layer Security，TLS）和安全 HTTP
286 （Secure HTTP，HTTPS）协议更为合适一些。

但摘要认证比它要取代的基本认证强大很多。与很多建议其他因特网服务使用的常用策略相比，（比如曾建议 LDAP、POP 和 IMAP 使用的 CRAM-MD5），摘要认证也要强大很多。

迄今为止，摘要认证还没有被广泛应用。但由于基本认证存在固有的安全风险，HTTP 设计者曾在 RFC 2617 中建议："在可行的情况下应该将目前在用的所有使用基本认证的服务，尽快地转换为摘要认证方式。"[3]这个标准的前景还不太明朗。

13.1.1 用摘要保护密码

摘要认证遵循的箴言是"绝不通过网络发送密码"。客户端不会发送密码，而是会发送一个"指纹"或密码的"摘要"，这是密码的不可逆扰码。客户端和服务器都知道这个密码，因此服务器可以验证所提供的摘要是否与密码相匹配。只拿到摘要的话，除了将所有的密码都拿来试试之外，没有其他方法可以找出摘要是来自哪个密

注 1：用户名和密码用 Base-64 编码进行了扰码，但很容易被解码。只能防止无意中的查看，没有任何防止恶意用户攻击的手段。

注 2：比如，与基于公有密钥的机制相比，摘要认证所提供的认证机制就不够强。同样，摘要认证除了能保护密码外，并没有提供保护其他内容的方式——请求和应答中的其余部分仍然可能被窃听。

注 3：随着 SSL 加密 HTTP 的流行和广泛采用，有关摘要认证的现实意义曾有过很激烈的争论。时间将会告诉我们摘要认证能否达到所需的规模。

码的！[4]

下面来看看摘要认证的工作原理（这是一个简化版本）。

- 在图 13-1a 中，客户端请求了某个受保护文档。
- 在图 13-1b 中，在客户端能够证明其知道密码从而确认其身份之前，服务器拒绝提供文档。服务器向客户端发起质询，询问用户名和摘要形式的密码。
- 在图 13-1c 中，客户端传递了密码的摘要，证明它是知道密码的。服务器知道所有用户的密码，[5]因此可以将客户提供的摘要与服务器自己计算得到的摘要进行比较，以验证用户是否知道密码。另一方在不知道密码的情况下，很难伪造出正确的摘要。
- 在图 13-1d 中，服务器将客户端提供的摘要与服务器内部计算出的摘要进行对比。如果匹配，就说明客户端知道密码（或者很幸运地猜中了！）。可以设置摘要函数，使其产生很多数字，让人不可能幸运地猜中摘要。服务器进行了匹配验证之后，会将文档提供给客户端——整个过程都没有在网络上发送密码。

287

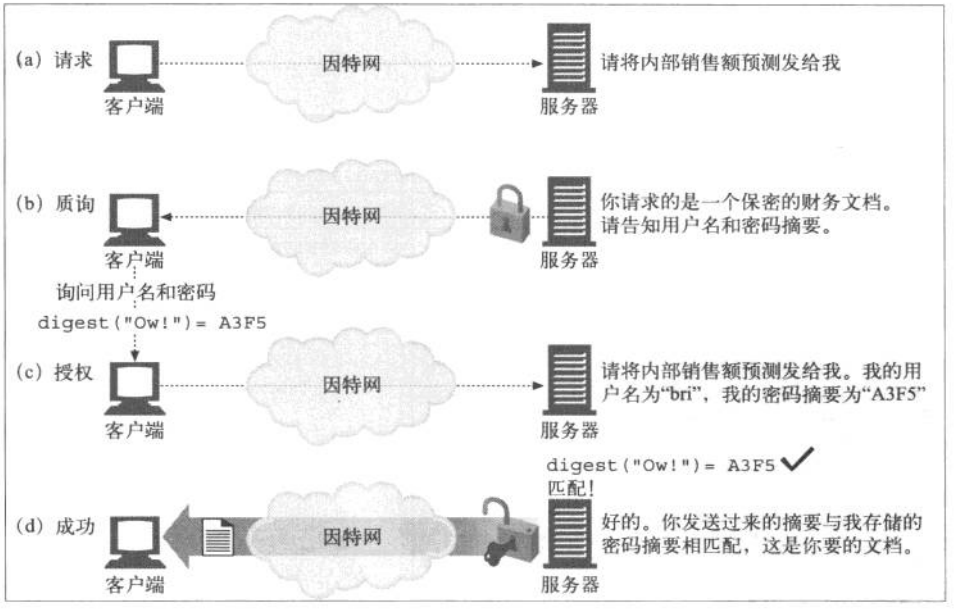

图 13-1　用摘要来实现隐藏密码的认证

注 4：有一些技术，比如词典攻击，会首先尝试一些常见的密码。这些密码分析技术可以极大地简化密码破译进程。

注 5：实际上，服务器只需要知道密码的摘要即可。

我们将在表 13-8 中更详细地讨论摘要认证中那些特殊的首部。

13.1.2 单向摘要

摘要是“对信息主体的浓缩”。[6] 摘要是一种单向函数，主要用于将无限的输入值转换为有限的浓缩输出值。[7] 常见的摘要函数 MD5，[8] 会将任意长度的字节序列转换为一个 128 位的摘要。

128 位 = 2^{128}，或者大约

288 1 000 000 000 000 000 000 000 000 000 000 000 000 种不同的输出值。

对这些摘要来说，最重要的是如果不知道密码的话，要想正确地猜出发送给服务器的摘要将是非常困难的。同样，如果有摘要，想要判断出它是由无数输入值中的哪一个产生的，也是非常困难的。

MD5 输出的 128 位的摘要通常会被写成 32 个十六进制的字符，每个字符表示 4 位。表 13-1 给出了几个示例输入的 MD5 摘要。注意 MD5 是怎样根据任意的输入产生定长的摘要输出的。

表13-1 MD5摘要实例

输　入	MD5摘要
"Hi"	C1A5298F939E87E8F962A5EDFC206918
"bri:Ow!"	BEAAA0E34EBDB072F8627C038AB211F8
"3.1415926535897"	475B977E19ECEE70835BC6DF46F4F6DE
"http://www.http-guide.com/index.htm"	C617C0C7D1D05F66F595E22A4B0EAAA5
"WE hold these Truths to be self-evident, that all Men are created equal, that they are endowed by their Creator with certain unalienable Rights, that among these are Life, Liberty and the Pursuit of Happiness—That to secure these Rights, Governments are instituted among Men, deriving their just Powers from the Consent of the Governed, that whenever any Form of Government becomes destructive of these Ends, it is the Right of the People to alter or to abolish it, and to institute new Government, laying its Foundation on such Principles, and organizing its Powers in such Form, as to them shall seem most likely to effect their Safety and Happiness."	66C4EF58DA7CB956BD04233FBB64E0A4

注 6：韦氏词典，1998 年。

注 7：理论上来讲，我们将数量无限的输入值转换成了数量有限的输出值，所以两个不同的输入值就可能映射为同一个摘要。这种情况被称为*冲突*（collision）。实际上，由于可用输出值的数量足够大，所以在现实生活中，出现冲突的可能是微乎其微的，对我们要实现的密码匹配来说并不重要。

注 8：MD5 表示“报文摘要的第五版”，是摘要算法系列中的一种。*安全散列算法*（Secure Hash Algorithm，SHA）是另一种常见的摘要函数。

有时也将摘要函数称为加密的校验和、单向散列函数或指纹函数。

13.1.3 用随机数防止重放攻击

使用单向摘要就无需以明文形式发送密码了。可以只发送密码的摘要，而且可以确信，没有哪个恶意用户能轻易地从摘要中解码出原始密码。

但是，仅仅隐藏密码并不能避免危险，因为即便不知道密码，别有用心的人也可以截获摘要，并一遍遍地重放给服务器。摘要和密码一样好用。

为防止此类重放攻击的发生，服务器可以向客户端发送一个称为随机数（nonce）[9]的特殊令牌，这个数会经常发生变化（可能是每毫秒，或者是每次认证都变化）。客户端在计算摘要之前要先将这个随机数令牌附加到密码上去。 289

在密码中加入随机数就会使摘要随着随机数的每一次变化而变化。记录下的密码摘要只对特定的随机值有效，而没有密码的话，攻击者就无法计算出正确的摘要，这样就可以防止重放攻击的发生。

摘要认证要求使用随机数，因为这个小小的重放弱点会使未随机化的摘要认证变得和基本认证一样脆弱。随机数是在 `WWW-Authenticate` 质询中从服务器传送给客户端的。

13.1.4 摘要认证的握手机制

HTTP 摘要认证协议是一种升级版的认证方式，所用首部与基本认证类似。它在传统首部中添加了一些新的选项，还添加了一个新的可选首部 `Authorization-Info`。

图 13-2 描述了简化的摘要认证三步握手机制。

图 13-2 中发生的情况如下所述。

- 在第（1）步中，服务器会计算出一个随机数。在第（2）步中，服务器将这个随机数放在 `WWW-Authenticate` 质询报文中，与服务器所支持的算法列表一同发往客户端。 290
- 在第（3）步中，客户端选择一个算法，计算出密码和其他数据的摘要。在第（4）步中，将摘要放在一条 `Authorization` 报文中发回服务器。如果客户端要对服务器进行认证，可以发送客户端随机数。
- 在第（5）步中，服务器接收摘要、选中的算法以及支撑数据，计算出与客户端相同的摘要。然后服务器将本地生成的摘要与网络传送过来的摘要进行比较，验

注 9：随机数这个词表示“本次”或“临时的”。在计算机安全的概念中，随机数捕获了一个特定的时间点，将其加入到安全计算之中。

证其是否匹配。如果客户端反过来用客户端随机数对服务器进行质询，就会创建客户端摘要。服务器可以预先将下一个随机数计算出来，提前将其传递给客户端，这样下一次客户端就可以预先发送正确的摘要了。

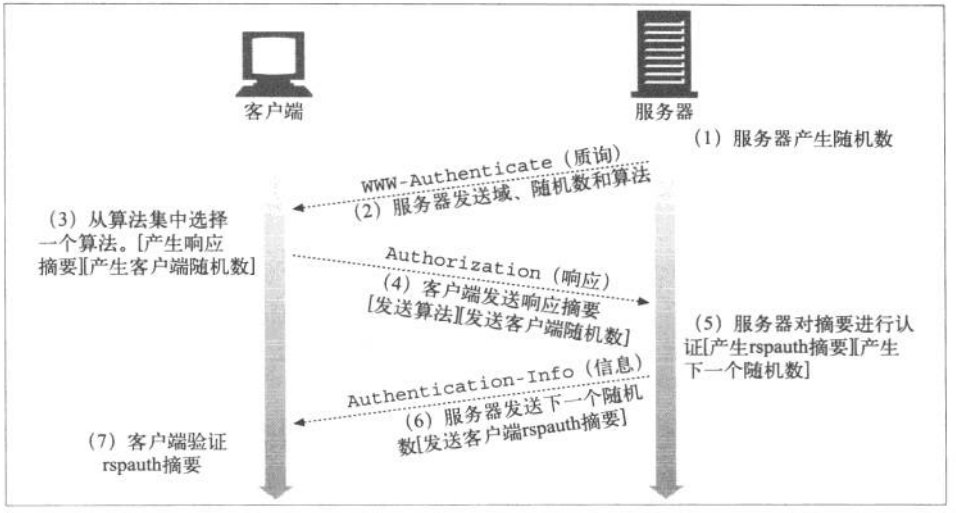

图 13-2 摘要认证的握手机制

这些信息中很多是可选的，而且有默认值。为了说明问题，图 13-3 对比了基本认证中发送的报文（参见图 13-3a 至图 13-3d）与简单的摘要认证实例发送的报文（参见图 13-3e 至图 13-3h）。

现在我们来更详细地探讨摘要认证的内部工作原理。

13.2 摘要的计算

摘要认证的核心就是对公共信息、保密信息和有时限的随机值这个组合的单向摘要。现在我们来看看这些摘要是如何计算出来的。摘要计算通常都是简单易懂的。[10] 附录 F 提供了示例源代码。

13.2.1 摘要算法的输入数据

摘要是根据以下三个组件计算出来的。

- 由单向散列函数 H(*d*) 和摘要 KD(*s*,*d*) 组成的一对函数，其中 *s* 表示密码，*d* 表示数据。

注 10：但对初学者来说，可选的 RFC 2617 兼容模式以及规范中背景资料的缺乏，使其变得有些复杂。我们会努力提供一些帮助。

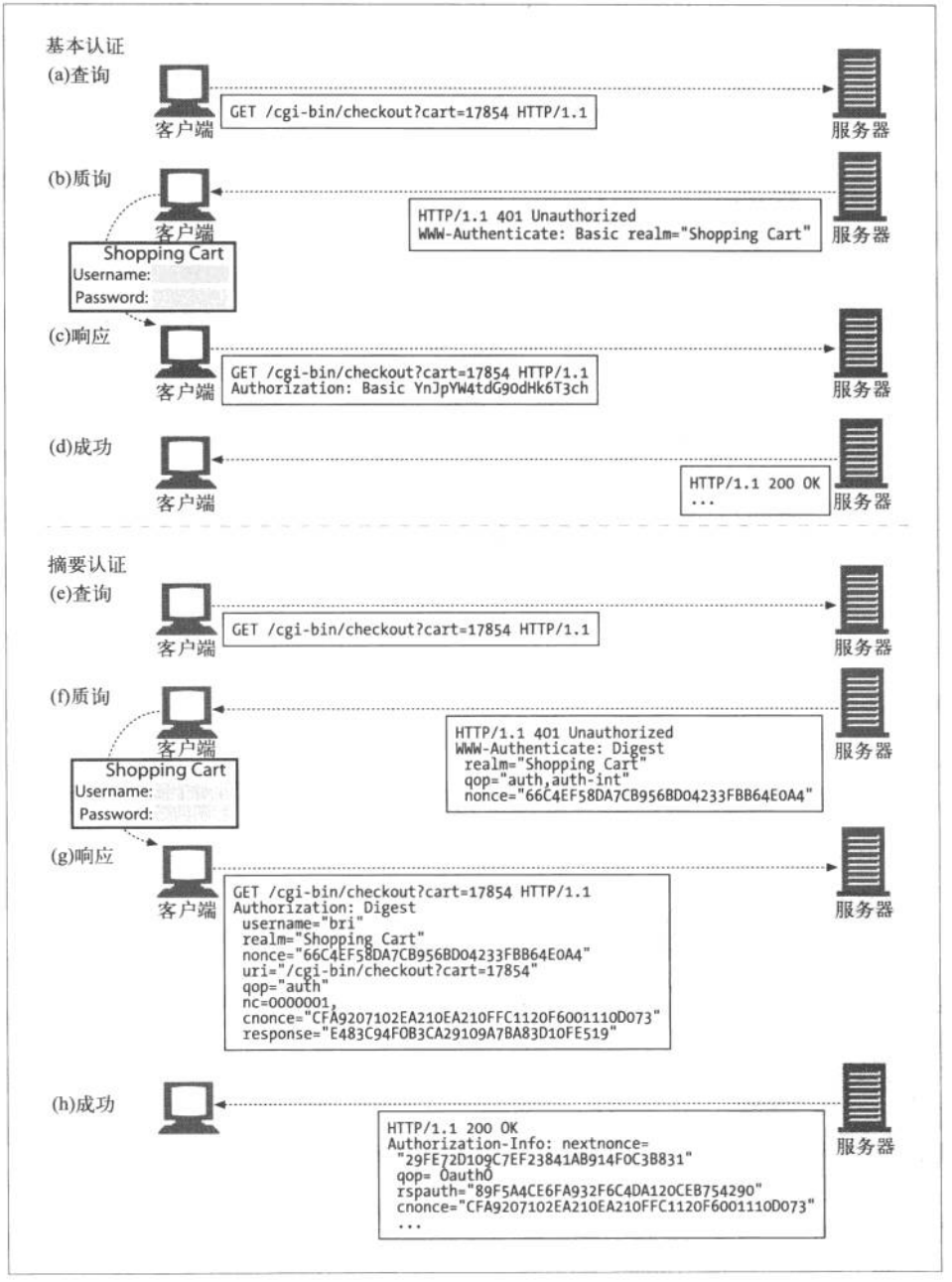

图 13-3　基本认证与摘要认证的语法对比

- 一个包含了安全信息的数据块，包括密码，称为 A1。
- 一个包含了请求报文中非保密属性的数据块，称为 A2。

H 和 KD 处理两块数据 A1 和 A2，产生摘要。

13.2.2 算法H(*d*)和KD(*s*,*d*)

291 ~ 292

摘要认证支持对各种摘要算法的选择。RFC 2617 建议的两种算法为 MD5 和 MD5-sess（“sess”表示会话），如果没有指定其他算法，默认算法为 MD5。

不管使用的是 MD5 还是 MD5-sess，都会用函数 H 来计算数据的 MD5，用摘要函数 KD 来计算以冒号连接的密码和非保密数据的 MD5。例如：

```
H(<data>) = MD5(<data>)
KD(<secret>,<data>) = H(concatenate(<secret>:<data>))
```

13.2.3 与安全性相关的数据（A1）

被称为 A1 的数据块是密码和受保护信息的产物，它包含有用户名、密码、保护域和随机数等内容。A1 只涉及安全信息，与底层报文自身无关。A1 会与 H、KD 和 A2 一同用于摘要计算。

RFC 2617 根据选择的算法定义了两种计算 A1 的方式。

- MD5

为每条请求运行单向散列函数。A1 是由冒号连接起来的用户名、域以及密码三元组。

- MD5-sess

只在第一次 `WWW-Authenticate` 握手时运行一次散列函数。对用户名、域和密码进行一次 CPU 密集型散列，并将其放在当前随机数和客户端随机数（cnonce）的前面。

表 13-2 显示了 A1 的定义。

表13-2 算法对A1的定义

算法	A1
MD5	`A1 = <user>:<realm>:<password>`
MD5-sess	`A1 = MD5(<user>:<realm>:<password>):<nonce>:<cnonce>`

13.2.4 与报文有关的数据（A2）

数据块 A2 表示的是与报文自身有关的信息，比如 URL、请求方法和报文实体的主

体部分。A2 有助于防止方法、资源或报文被篡改。A2 会与 H、KD 和 A1 一起用于摘要的计算。

RFC 2617 根据所选择的保护质量（qop），为 A2 定义了两种策略。

- 第一种策略只包含 HTTP 请求方法和 URL。当 qop="auth" 时使用这种策略，这是默认的情况。
- 第二种策略添加了报文实体的主体部分，以提供一定程度的报文完整性检测。qop="auth-int" 时使用。

293

表 13-3 显示了 A2 的定义。

表13-3 算法对A2的定义（请求摘要）

qop	A2
未定义	<request-method>:<uri-directive-value>
auth	<request-method>:<uri-directive-value>
auth-int	<request-method>:<uri-directive-value>:H(<request-entity-body>)

request-method 是 HTTP 的请求方法。uri-directive-value 是请求行中的请求 URI。可能是个 "*"、absoluteURL 或者 abs_path，但它必须与请求 URI 一致。尤其需要注意的是，如果请求 URI 是 absoluteURL，它必须是个绝对 URL。

13.2.5 摘要算法总述

RFC 2617 定义了两种给定了 H、KD、A1 和 A2 之后，计算摘要的方式。

- 第一种方式要与老规范 RFC 2069 兼容，在没有 qop 选项的时候使用。它是用保密信息和随机报文数据的散列值来计算摘要的。
- 第二种方式是现在推荐使用的方式——这种方式包含了对随机数计算和对称认证的支持。只要 qop 为 auth 或 auth-int，就要使用这种方式。它向摘要中添加了随机计数、qop 和 cnonce 数据。

表 13-4 给出了得到的摘要函数定义。注意得到的摘要使用了 H、KD、A1 和 A2。

表13-4 新/老摘要算法

qop	摘要算法	备 注
未定义	KD(H(A1), <nonce>:H(A2))	不推荐
auth 或 auth-int	KD(H(A1), <nonce>:<nc>:<cnonce>:<qop>:H(A2))	推荐

这些派生封装层很容易把人弄晕。这也是有些读者觉得 RFC 2617 难懂的原因之一。为了简化，表 13-5 扩展了 H 和 KD 的定义，用 A1 和 A2 来表示摘要。

表13-5 展开的摘要算法备忘单

qop	算 法	展开的算法
未定义	<undefined>	MD5(MD5(A1):<nonce>:MD5(A2))
	MD5	
	MD5-sess	
auth	<undefined>	MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))
	MD5	
	MD5-sess	
auth-int	<undefined>	MD5(MD5(A1):<nonce>:<nc>:<cnonce>:<qop>:MD5(A2))
	MD5	
	MD5-sess	

13.2.6 摘要认证会话

客户端响应对保护空间的 WWW-Authenticate 质询时，会启动一个此保护空间的认证会话（与受访问服务器的标准根结合在一起的域就定义了一个“保护空间”）。

在客户端收到另一条来自保护空间的任意一台服务器的 WWW-Authenticate 质询之前，认证会话会一直持续。客户端应该记住用户名、密码、随机数、随机数计数以及一些与认证会话有关的隐晦值，以便将来在此保护空间中构建请求的 Authorization 首部时使用。

随机数过期时，即便老的 Authorization 首部所包含的随机数不再新鲜了，服务器也可以选择接受其中的信息。服务器也可以返回一个带有新随机数的 401 响应，让客户端重试这条请求；指定这个响应为 stale=true，表示服务器在告知客户端用新的随机数来重试，而不再重新提示输入新的用户名和密码了。

13.2.7 预授权

在普通的认证方式中，事务结束之前，每条请求都要有一次请求 / 质询的循环，参见图 13-4a。

如果客户端事先知道下一个随机数是什么，就可以取消这个请求 / 质询循环，这样客户端就可以在服务器发出请求之前，生成正确的 Authorization 首部了。如果客户端能在服务器要求它计算 Authorization 首部之前将其计算出来，就可以预先将 Authorization 首部发送给服务器，而不用进行请求 / 质询了。图 13-4b 显示了这种方式对性能的影响。

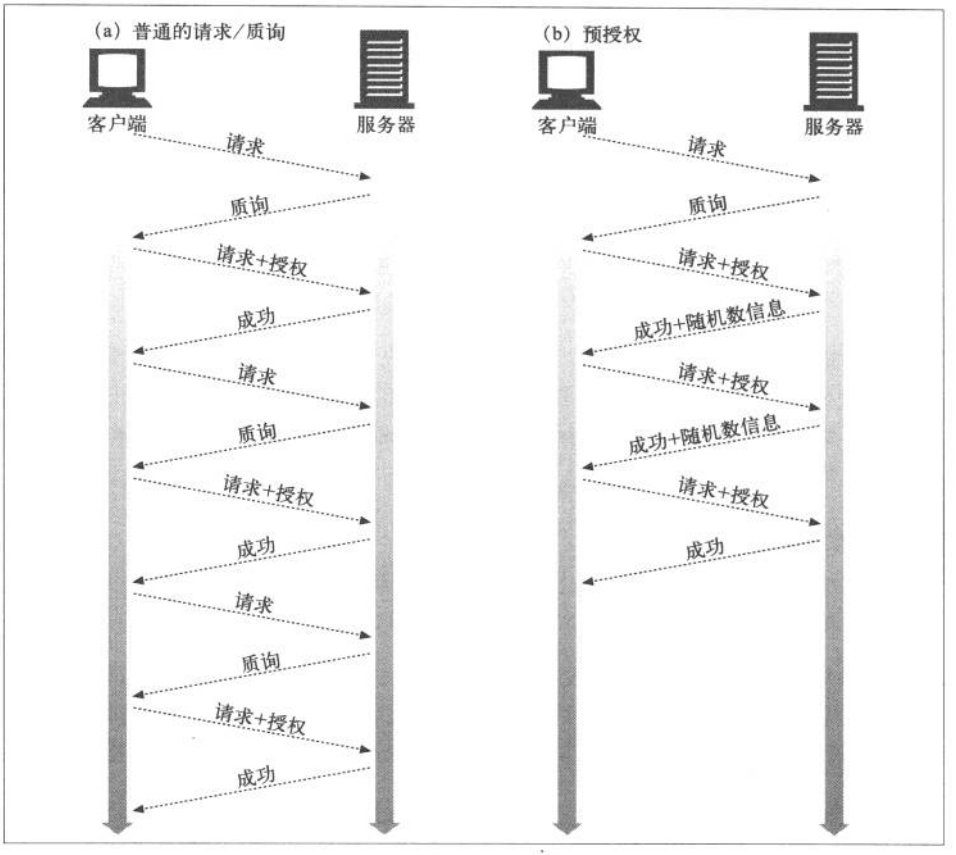

图 13-4　预授权减少了报文的数量

预授权对基本认证来说并不重要（而且很常见）。浏览器通常会维护一些客户端数据库以存储用户名和密码。一旦用户与某站点进行了认证，浏览器通常会为后继对那个 URL 的请求发送正确的 Authorization 首部（参见第 12 章）。 295

由于摘要认证使用了随机数技术来破坏重放攻击，所以对摘要认证来说，预授权要稍微复杂一些。服务器会产生任意的随机数，所以在客户端收到质询之前，不一定总能判定应该发送什么样的 Authorization 首部。

摘要认证在保留了很多安全特性的同时，还提供了几种预授权方式。这里列出了三种可选的方式，通过这些方式，客户端无需等待新的 WWW-Authenticate 质询，就可以获得正确的随机数：

- 服务器预先在 Authentication-Info 成功首部中发送下一个随机数；

- 服务器允许在一小段时间内使用同一个随机数；
296 - 客户端和服务器使用同步的、可预测的随机数生成算法。

1. 预先生成下一个随机数

可以在 Authentication-Info 成功首部中将下一个随机数预先提供给客户端。这个首部是与前一次成功认证的 200 OK 响应一同发送的。

```
Authentication-Info: nextnonce="<nonce-value>"
```

有了下一个随机数，客户端就可以预先发布 Authorization 首部了。

尽管这种预授权机制避免了请求 / 质询循环（加快了事务处理的速度），但实际上它也破坏了对同一台服务器的多条请求进行管道化的功能，因为在发布下一条请求之前，一定要收到下一个随机值才行。而管道化是避免延迟的一项基本技术，所以这样可能会造成很大的性能损失。

2. 受限的随机数重用机制

另一种方法不是预先生成随机数序列，而是在有限的次数内重用随机数。比如，服务器可能允许将某个随机数重用 5 次，或者重用 10 秒。

在这种情况下，客户端可以随意发布带有 Authorization 首部的请求，而且由于随机数是事先知道的，所以还可以对请求进行管道化。随机数过期时，服务器要向客户端发送 401 Unauthorized 质询，并设置 WWW-Authenticate:stale=true 指令：

```
WWW-Authenticate: Digest
    realm="<realm-value>"
    nonce="<nonce-value>"
    stale=true
```

重用随机数使得攻击者更容易成功地实行重放攻击。虽然这确实降低了安全性，但重用的随机数的生存期是可控的（从严格禁止重用到较长时间的重用），所以应该可以在安全和性能间找到平衡。

此外，还可以通过其他一些特性使重放攻击变得更加困难，其中就包括增量计数器和 IP 地址测试。但这些技术只能使攻击的实施更加麻烦，并不能消除由此带来的安全隐患。

3. 同步生成随机数

还可以采用时间同步的随机数生成算法，客户端和服务器可根据共享的密钥，生成第三方无法轻易预测的、相同的随机数序列（比如安全 ID 卡）。

297 这些算法都超出了摘要认证规范的范畴。

13.2.8 随机数的选择

随机数的内容不透明，而且与实现有关。但性能、安全性和便捷性的优劣都取决于明智的选择。

RFC 2617 建议采用这个假想的随机数公式：

```
BASE64(time-stamp H(time-stamp ":" ETag ":" private-key))
```

其中 time-stamp 是服务器产生的时间或其他不会重复的值，ETag 是与所请求实体有关的 HTTP ETag 首部的值，private-key 是只有服务器知道的数据。

有了这种形式的随机数，服务器就可以在收到客户端的认证首部之后重新计算散列部分，如果结果与那个首部的随机数不符，或者时间戳的值不够新，就拒绝请求。服务器可以通过这种方式来限制随机数的有效持续时间。

包含 Etag 可以防止对已更新资源版本的重放请求。（注意，在随机数中包含客户端的 IP 地址，服务器好像就可以限制原来获得此随机数的客户端重用这个随机数了，但这会破坏代理集群的工作。使用代理集群时，来自单个用户的多条请求通常会经过不同的代理进行传输，而且 IP 地址欺骗实现起来也不是很难。）

实现可以选择不接受以前使用过的随机数或摘要，以防止重放攻击。实现也可以选择为 POST 或 PUT 请求使用一次性的随机数或摘要，为 GET 请求使用时间戳。

会影响到随机数选取的一些实际安全问题参见 13.5 节。

13.2.9 对称认证

RFC 2617 扩展了摘要认证机制，允许客户端对服务器进行认证。这是通过提供客户端随机值来实现的，服务器会根据它对共享保密信息的正确了解生成正确的响应摘要。然后，服务器在 Authorization-Info 首部中将此摘要返回给客户端。

这种对称认证方式被标准化为 RFC 2617。为了与原有 RFC 2069 标准后向兼容，它是可选的，但由于它提供了一些重要的安全提升机制，强烈推荐现今所有的客户端和服务器都要实现全部 RFC 2617 特性。特别是，只要提供了 qop 指令，就要求执行对称认证，而没有 qop 指令时则不要求执行对称认证。

响应摘要的计算方法与请求摘要类似，但由于响应中没有方法，而且报文实体数据有所不同，所以只有报文主体信息 A2 不同。表 13-6 和表 13-7 对比了请求和响应摘要中 A2 的计算方法。 298

表13-6 算法中A2的定义（请求摘要）

qop	A2
未定义	<request-method>:<uri-directive-value>
auth	<request-method>:<uri-directive-value>
auth-int	<request-method>:<uri-directive-value>:H(<request-entity-body>)

表13-7 算法中A2的定义（响应摘要）

qop	A2
未定义	:<uri-directive-value>
auth	:<uri-directive-value>
auth-int	:<uri-directive-value>:H(<response-entity-body>)

cnonce 值和 nc 值必须是本报文所响应的客户端请求中的相应值。如果指定了 qop="auth" 或 qop="auth-int"，就必须提供响应 auth、cnonce 和 nonce 计数指令。

13.3 增强保护质量

可以在三种摘要首部中提供 qop 字段：WWW-Authenticate、Authorization 和 Authentication-Info。

通过 qop 字段，客户端和服务器可以对不同类型及质量的保护进行协商。比如，即便会严重降低传输速度，有些事务可能也要检查报文主体的完整性。

服务器首先在 WWW-Authenticate 首部输出由逗号分隔的 qop 选项列表。然后客户端从中选择一个它支持且满足其需求的选项，并将其放在 Authorization 的 qop 字段中回送给服务器。

qop 字段是可选的，但只是在后向兼容原有 RFC 2069 规范的情况下才是可选的。现代所有的摘要实现都应该支持 qop 选项。

RFC 2617 定义了两种保护质量的初始值：表示认证的 auth，带有报文完整性保护的认证 auth-int。将来可能还会出现其他 qop 选项。

13.3.1 报文完整性保护

如果使用了完整性保护（qop="auth-int"），H（实体的主体部分）就是对实体主体部分，而不是报文主体部分的散列。对于发送者，要在应用任意传输编码方式之前计算；而对于接收者，则应在去除所有传输编码之后计算。注意，对于任何含有多部份的内容类型来说，多部分的边界和每部分中嵌入的首部都要包含在内。

299

13.3.2 摘要认证首部

基本认证和摘要认证协议都包含了WWW-Authenticate首部承载的授权质询、Authorization首部承载的授权响应。摘要认证还添加了可选的Authorization-Info首部，这个首部是在成功认证之后发送的，用于实现三步握手机制，并传送下一个随机数。表13-8给出了基本认证和摘要认证的首部。

表13-8 HTTP认证首部

阶段	基 本	摘 要
质询	WWW-Authenticate: Basic realm="<realm-value>"	WWW-Authenticate: Digest realm="<realm-value>" nonce="<nonce-value>" [domain="<list-of-URIs>"] [opaque="<opaque-token-value>"] [stale=<true-or-false>] [algorithm=<digest-algorithm>] [qop="<list-of-qop-values>"] [<extension-directive>]
响应	Authorization: Basic <base64(user:pass)>	Authorization: Digest username="<username>" realm="<realm-value>" nonce="<nonce-value>" uri=<request-uri> response="<32-hex-digit-digest>" [algorithm=<digest-algorithm>] [opaque="<opaque-token-value>"] [cnonce="<nonce-value>"] [qop=<qop-value>] [nc=<8-hex-digit-nonce-count>] [<extension-directive>]
Info	n/a	Authentication-Info: nextnonce="<nonce-value>" [qop="<list-of-qop-values>"] [rspauth="<hex-digest>"] [cnonce="<nonce-value>"] [nc=<8-hex-digit-nonce-count>]

摘要认证首部要复杂得多。详细介绍参见附录F。

13.4 应该考虑的实际问题

使用摘要认证时需要考虑几件事情。本节讨论了其中一些问题。 300

13.4.1 多重质询

服务器可以对某个资源发起多重质询。比如，如果服务器不了解客户端的能力，就可以既提供基本认证质询，又提供摘要认证质询。客户端面对多重质询时，必须以它所支持的最强的质询机制来应答。

质询自身可能会包含由逗号分隔的认证参数列表。如果 WWW-Authenticate 或 Proxy-Authenticate 首部包含了多个质询，或者提供了多个 WWW-Authenticate 首部，用户 Agent 代理在解析 WWW-Authenticate 或 Proxy-Authenticate 首部字段值时就要特别小心。注意，很多浏览器只支持基本认证，要求这是提交给它的第一种认证机制。

在提供了认证选项范围的情况下，安全问题上就会存在明显的“最薄弱环节”。只有当基本认证是最低可接受认证方式时，服务器才应该包含它，而且管理员还应该警告用户，即使运行了不同层次安全措施，系统间使用相同密码也存在一定危险性。

13.4.2 差错处理

在摘要认证中，如果某个指令或其值使用不当，或者缺少某个必要指令，就应该使用响应 400 Bad Request。

如果请求的摘要不匹配，就应该记录一次登录失败。某客户端连续多次失败可能说明有攻击者正在猜测密码。

认证服务器一定要确保 URI 指令指定的资源与请求行中指定的资源相同。如果不同，服务器就应该返回 400 Bad Request 错误。（这可能是一种攻击的迹象，因此服务器设计者可能会考虑将此类错误记录下来。）这个字段包含的内容与请求 URL 中的内容是重复的，用来应对中间代理可能对客户端请求进行的修改。这个经过修改（但估计语义是等价的）的请求计算后得到的摘要可能会与客户端计算出的摘要有所不同。

13.4.3 保护空间

域值，与被访问服务器的标准根 URL 结合在一起，定义了保护空间。

通过域可以将服务器上的受保护资源划分为一组保护空间，每个空间都有自己的认
301 证机制和 / 或授权数据库。域值是一个字符串，通常由原始服务器分配，可能会有认证方案特有的附加语义。注意，可能会有多个授权方案相同，而域不同的质询。

保护空间确定了可以自动应用证书的区域。如果前面的某条请求已被授权，在一段

时间内，该保护空间中所有其他请求都可以重用同一个证书，时间的长短由认证方案、参数和/或用户喜好来决定。除非认证方案进行了其他定义，否则单个保护空间是不能扩展到其服务器范围之外的。

对保护空间的具体计算取决于认证机制。

- 在基本认证中，客户端会假定请求URI中或其下的所有路径都与当前的质询处于同一个保护空间内。客户端可以预先提交对此空间中资源的认证，无需等待来自服务器的另一条质询。
- 在摘要认证中，质询的WWW-Authenticate:domain字段对保护空间作了更精确的定义。domain字段是一个用引号括起来的、中间由空格分隔的URI列表。通常认为，domain列表中的所有URI和逻辑上处于这些前缀之下的所有URI，都位于同一个保护空间中。如果没有domain字段，或者此字段为空，质询服务器上的所有URI就都在保护空间内。

13.4.4 重写URI

代理可以通过改变URI语法，而不改变所描述的实际资源的方式来重写URI。比如：

- 可以对主机名进行标准化，或用IP地址来取代；
- 可以用“%”转义形式来取代嵌入的字符；
- 如果某类型的一些附加属性不会影响从特定原始服务器上获取资源，就可以将其附加或插入到URI中。

代理可修改URI，而且摘要认证会检查URI值的完整性，所以如果进行了任意一种修改，摘要认证就会被破坏。更多信息参见13.2.4节。

13.4.5 缓存

共享的缓存收到包含Authorization首部的请求和转接那条请求产生的响应时，除非响应中提供了下列两种Cache-Control指令之一，否则一定不能将那条响应作为对任何其他请求的应答使用。

- 如果原始响应中包含有Cache-Control指令must-revalidate，缓存可以在应答后继请求时使用那条响应的实体部分。但它首先要用新请求的请求首部，与原始服务器再次进行验证，这样原始服务器就可以对新请求进行认证了。
- 如果原始响应中包含有Cache-Control指令public，在对任意后继请求的应答中都可以返回响应的实体部分。 302

13.5 安全性考虑

RFC 2617 总结了 HTTP 认证方案固有的一些安全风险，这是很令人钦佩的。本节描述了其中的部分风险。

13.5.1 首部篡改

为了提供一个简单明了的防首部篡改系统，要么就得进行端到端的加密，要么就得对首部进行数字签名——最好是两者的结合！摘要认证的重点在于提供一种防篡改认证机制，但并不一定要将这种保护扩展到数据上去。具有一定保护级别的首部只有 WWW-Authenticate 和 Authorization。

13.5.2 重放攻击

在当前的上下文中，重放攻击指的就是有人将从某个事务中窃取的认证证书用于另一个事务。尽管对 GET 请求来说这也是个问题，但为 POST 和 PUT 请求提供一种简单的方式来避免重放攻击才是非常必要的。在传输表单数据的同时，成功重放原先用过的证书会引发严重的安全问题。

因此，为了使服务器能够接受“重放的”证书，还必须重复发送随机数。缓解这个问题的方法之一就是让服务器产生的随机数包含（如前所述）根据客户端 IP 地址、时间戳、资源 Etag 和私有服务器密钥算出的摘要。这样，IP 地址和一个短小超时值的组合就会给攻击者造成很大的障碍。

但这种解决方案有一个很重要的缺陷。我们之前讨论过，用客户端 IP 地址来创建随机数会破坏经过代理集群的传输。在这类传输中。来自单个用户的多条请求可能会穿过不同的代理。而且，IP 欺骗也并不难实现。

一种可以完全避免重放攻击的方法就是为每个事务都使用一个唯一的随机数。在这种实现方式中，服务器会为每个事务发布唯一的随机数和一个超时值。发布的随机数只对指定的事务有效，而且只在超时值的持续区间内有效。这种方式会增加服务器的负担，但这种负担可忽略不计。

13.5.3 多重认证机制

服务器支持多重认证机制（比如基本认证和摘要认证）时，通常会在 WWW-Authenticate
303 首部提供选项。由于没有要求客户端选择功能最强的认证机制，所以得到的认证效果就和功能最弱的认证方案差不多。

要避免出现这个问题，最直接的方法就是让客户端总是去选择可用认证方案中功能

最强的那个。如果无法实现（因为大部分人使用的都是商业化客户端），唯一的选择就是使用一个只维护最强认证方案的代理服务器。但只有在已知所有客户端都支持所选认证方案的区域中才能采用这种方式——比如，在公司网络中。

13.5.4 词典攻击

词典攻击是典型的密码猜测型攻击方式。恶意用户对某个事务进行窃听，并对随机数 / 响应对使用标准的密码猜测程序。如果用户使用的是相对比较简单的密码，而且服务器使用的也是简单的随机数，它很可能会找到匹配项。如果没有密码过期策略，只要有足够的时间和破解密码所需的一次性费用，就很容易搜集到足够多的密码，造成实质性的破坏。

除了使用复杂的相对难以破译的密码和合适的密码过期策略之外，确实没有什么好的方法可以解决这个问题。

13.5.5 恶意代理攻击和中间人攻击

现在很多因特网流量都会在这个或那个地方流经某个代理。随着重定向技术和拦截代理的出现，用户甚至都意识不到他的请求穿过了某个代理。如果这些代理中有一个是恶意的或者容易被入侵的，就会使客户端置于中间人攻击之下。

这种攻击可以采用窃听的形式，也可以删除提供的所有选项，用最薄弱的认证策略（比如基本认证）来取代现有的认证机制，对其进行修改。

入侵受信代理的方式之一就是使用其扩展接口。有时代理会提供复杂的编程接口，可以为这类代理编写一个扩展（比如，plug-in）来拦截流量并对其进行修改。不过，数据中心和代理自身提供的安全性使得通过恶意 plug-in 进行中间人攻击的可能性变得很渺茫。

没有什么好办法可以解决这个问题。可行的解决方案包括由客户端提供与认证功能有关的可见线索，对客户端进行配置使其总是使用可用认证策略中功能最强的那一种，等等。但即使使用的是最强大的认证策略，客户端仍然很容易被窃听。防止这些攻击唯一简便的方式就是使用 SSL。 304

13.5.6 选择明文攻击

使用摘要认证的客户端会用服务器提供的随机数来生成响应。但如果中间有一个被入侵的或恶意的代理在拦截流量（或者有个恶意的原始服务器），就可以很容易地为客户端的响应计算提供随机数。使用已知密钥来计算响应可以简化响应的密码分析过程。这种方式被称为*选择明文攻击*（chosen plaintext attack）。选择明文攻击有以下几种变体形式。

- 预先计算的词典攻击

  这是词典攻击和选择明文攻击的组合。首先，发起攻击的服务器会用预先确定的随机数和常见密码的变化形式产生一组响应，创建一个词典。一旦有了规模可观的词典，攻击服务器或代理就可以完成对流量的封锁，向客户端发送预先确定的随机数。攻击者从客户端得到一个响应时，会搜索生成的词典，寻找匹配项。如果有匹配项，攻击者就捕获了这个用户的密码。

- 批量暴力型攻击

  批量暴力型攻击的不同之处在于计算密码的方式。它没有试图去匹配预先计算出来的摘要，而是用一组机器枚举了指定空间内所有可能的密码。随着机器运行速度变得越来越快，暴力型攻击的可行性也变得越来越强了。

总之，这些攻击所造成的威胁是很容易应对的。防止这些攻击的一种方法就是配置客户端使用可选的 cnonce 指令，这样响应就是基于客户端的判断产生的，而不是用服务器提供的随机数（这个随机数可能会被攻击者入侵）产生的。通过这种方法，再结合一些强制使用合理强密码的策略，以及一个好的密码过期策略，就可以完全消除选择明文攻击的威胁。

13.5.7 存储密码

摘要认证机制将对比用户的响应与服务器内部存储的内容——通常就是用户名和 H(A1) 元组对，其中 H(A1) 是从用户名、域和密码的摘要中导出的。

与 Unix 机器中传统的密码文件不同，如果摘要认证密码文件被入侵了，攻击者马上就能够使用域中所有文件，不需要再进行解码了。

消除这个问题的方法包括：

- 就像密码文件中包含的是明文密码一样来保护它；
- 确保域名在所有域中是唯一的。这样，如果密码文件被入侵，所造成的破坏也只
305 局限于一个特定的域中。包含主机和 domain 的全路径域名就可以满足这个要求。

尽管摘要认证提供的解决方案比基本认证要强壮且安全得多，但它并没有为内容的安全提供任何保证——真正安全的事务只有通过 SSL 才能实现，我们将在下一章介绍。

13.6 更多信息

更多有关认证的信息，参见以下资源。

- http://www.ietf.org/rfc/rfc2617.txt

RFC 2617，“HTTP Authentication：Basic and Digest Access Authentication.”
306 （“HTTP 认证：基本和摘要访问认证”）。