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