Google账户两步验证的工作原理

Dec 28, 2011 by Tao

我们往往会在不同的网站上使用相同的密码，这样一旦一个网站账户的密码泄露，就会危及到其他使用相同密码的账户的安全，这也是最近的密码泄露事件造成如此大影响的原因。为了解决这个问题，一些网站在登录时要求除了输入账户密码之外，还需要输入另一个一次性密码。银行常用的动态口令卡就是这种一次性密码的例子，在线支付网站的一次性短信密码则是另一种实现。

Google现在也推荐用户启用两步验证（Two-step verification）功能（Youtube上的视频介绍），并且除了以短信或者电话的方式发送一次性密码之外，还提供了另一种基于时间的一次性密码（Time-based One-time Password，简称TOTP），只需要在手机上安装密码生成应用程序，就可以生成一个随着时间变化的一次性密码，用于帐户验证，而且这个应用程序不需要连接网络即可工作。仔细看了看这个方案的实现原理，发现挺有意思的。下面简单介绍一下。

Google的两步验证算法源自另一种名为HMAC-Based One-Time Password的算法，简称HOTP。HOTP的工作原理如下：

客户端和服务器事先协商好一个密钥K，用于一次性密码的生成过程，此密钥不被任何第三方所知道。此外，客户端和服务器各有一个计数器C，并且事先将计数值同步。

进行验证时，客户端对密钥和计数器的组合(K,C)使用HMAC（Hash-based Message Authentication Code）算法计算一次性密码，公式如下：

```
HOTP(K,C) = Truncate(HMAC-SHA-1(K,C))
```

上面采用了HMAC-SHA-1，当然也可以使用HMAC-MD5等。HMAC算法得出的值位数比较多，不方便用户输入，因此需要截断（Truncate）成为一组不太长十进制数（例如6位）。计算完成之后客户端计数器C计数值加1。用户将这一组十进制数输入并且提交之后，服务器端同样的计算，并且与用户提交的数值比较，如果相同，则验证通过，服务器端将计数值C增加1。如果不相同，则验证失败。

这里的一个比较有趣的问题是，如果验证失败或者客户端不小心多进行了一次生成密码操作，那么服务器和客户端之间的计数器C将不再同步，因此需要有一个重新同步（Resynchronization）的机制。这里不作具体介绍，详情可以参看RFC 4226。

介绍完了HOTP，Time-based One-time Password（TOTP）也就容易理解了。TOTP将HOTP中的计数器C用当前时间T来替代，于是就得到了随着时间变化的一次性密码。非常有趣吧！

虽然原理很简单，但是用时间来替代计数器会有一些特殊的问题，这些问题也很有意思，我们选取几个进行一下探讨。

首先，时间T的值怎么选取？因为时间每时每刻都在变化，如果选择一个变化太快的T（例如从某一时间点开始的秒数），那么用户来不及输入密码。如果选择一个变化太慢的T（例如从某一时间点开始的小时数），那么第三方攻击者就有充足的时间去尝试所有可能的一次性密码（试想6位数字的一次性密码仅仅有10^6种组合），降低了密码的安全性。除此之外，变化太慢的T还会导致另一个问题。如果用户需要在短时间内两次登录账户，由于密码是一次性的不可重用，用户必须等到下一个一次性密码被生成时才能登录，这意味着最多需要等待59分59秒！这显然不可接受。综合以上考虑，Google选择了30秒作为时间片，T的数值为从Unix epoch（1970年1月1日00:00:00）来经历的30秒的个数。

第二个问题是，由于网络延时，用户输入延迟等因素，可能当服务器端接收到一次性密码时，T的数值已经改变，这样就会导致服务器计算的一次性密码值与用户输入的不同，验证失败。解决这个问题个一个方法是，服务器计算当前时间片以及前面的n个时间片内的一次性密码值，只要其中有一个与用户输入的密码相同，则验证通过。当然，n不能太大，否则会降低安全性。

事实上，这个方法还有一个另外的功能。我们知道如果客户端和服务器的时钟有偏差，会造成与上面类似的问题，也就是客户端生成的密码和服务端生成的密码不一致。但是，如果服务器通过计算前n个时间片的密码并且成功验证之后，服务器就知道了客户端的时钟偏差。因此，下一次验证时，服务器就可以直接将偏差考虑在内进行计算，而不需要进行n次计算。

以上就是Google两步验证的工作原理，推荐大家使用，这确实是保护帐户安全的利器。

参考资料

1. TOTP: Time-based One-time Password Algorithm, RFC Draft, http://tools.ietf.org/id/draft-mraihi-totp-timebased-06.html
2. HOTP: An HMAC-Based One-Time Password Algorithm, RFC 4226, http://tools.ietf.org/html/rfc4226

3. Google Authenticator project, http://code.google.com/p/google-authenticator/

Tags: google , security