面向用户的现代口令安全性指南

以用户为中心，介绍关于口令创建与保存的具体建议

作者：Ian Maddox与Kyle Moschetto，Google云解决方案架构师

本份白皮书主要面向需要使用身份验证功能的在线应用用户，内容涵盖口令功能指南与使用建议。本白皮书以用户为中心，详尽介绍关于口令创建与保存的具体建议，包括如何在口令强度与可用性之间做出权衡。如果您身为工程师并希望了解如何将身份验证机制引入在线应用程序，请参阅我们《面向系统设计师的口令安全》一文。

从计算技术诞生早期，业界就一直努力改进口令保护能力。完全开放授权共享显然不太现实，毕竟信息很可能因此落入恶意人士之手、或者被使用者所遗忘。由于缺乏对现实世界中安全用例的支持能力，加上最终用户糟糕的安全使用习惯，系统正面临愈发严峻的安全保障挑战。

最佳实践快速参考

要	不要
使用口令管理器。	将口令写在便笺上，或者以未加密形式保存。
为各个站点及应用程序设定唯一的口令。	在多个站点上重复使用同一口令。
设置较长且随机的口令内容。	使用较短的口令或者易被猜到的替换方式（例如用@代替字母a）。
使用多词口令短语。	使用单一词语（例如「password」）
尽可能使用单点登录机制。	在同一服务上创建多个账户。
在安全问题中使用加盐（salted）口令短语或算法。	不加保护、直接按实际情况设置安全提示问题。
信任并参考关于安全最佳实践的权威信息。	坚持采用十年之前的Web安全实践。

概述

本白皮书主要涵盖以下内容：

- 在口令安全方面，提供经过事实验证及研究支持的可靠信息来源
- 关于最终用户口令的安全处理建议
- 关于口令安全性问题的常见错误与误解

- 介绍其他相关技术

术语

熵

口令相关术语，用于描述口令的不可预测性，通常以位（bit）为单位表示。

哈希

由单向且在数学上不可逆的确定性加密算法得出的结果。

MFA与2FA

全称为多因素验证与双因素验证。指在传统口令之外作为补充的其他验证方法。

口令**管理器**

用于创建、存储以及检索口令的软件。

彩虹表

由系统预先计算得出的哈希表及其明文值，通常用于将哈希与口令或者口令短语相匹配。

加盐）

向输入中添加随机数据，通常用于在原始输入内生成不可预测的变体哈希，进而破坏彩虹表。

单点登录

也被称为联合认证。这是一套可供网站或服务通过可信第三方实现身份验证的系统，例如用户可利用Google账户登录至其他应用程序或者社交媒体网站。

本指南适用场景

本指南涵盖多种利用口令控制云原生资源访问的选项。受篇幅所限，本指南不可能涵盖所有使用场景，也并不提供特定解决方案。我们的目标是帮助最终用户、安全专业人员、系统设计师以及云架构师提高口令管理意识。

2019 年口令安全状况概述

口令安全性是个庞大而广泛的主题，其中自然涉及多种多样的问题与争议。不同组织往往会在何谓安全行为、何谓不安全行为之间划分出自己的界限。本指南的出发点并非为各类应用程序提供明确的口令使用指导，而是概述用户群体在使用口令系统时应当考虑的核心问题。每款应用程序对于安全状态的要求取决于应用数据安全等级，因此需要一事一议、不宜粗暴推广。

口令无处不在——作为当前最为常见的验证机制，口令的应用场景涵盖一切垂直市场、行业乃至应用程序类别。与其他常见的安全功能不同，口令始终独立存在，且口令行业本身一直没有找到统一的共识性实现、管理与存储方法。

之所以缺乏广泛共识，主要原因是安全领域的发展过于迅速。我们当下认为安全的机制，可能在明天就快速过时、遭到入侵破解，甚至彻底被时代所抛弃。当前被广泛认可的尖端技术，很快就会被新技术、新工具以及新程序所取代。因此，用户与系统设计人员必须始终紧跟口令安全领域的变化与发展趋势。

权威来源

关于口令使用的最新指南，目前市面上有着不少可信来源。以下清单当然无法全部涵盖，但我们会尽可能提及其中最重要的几项：

- 美国国家标准技术研究院（NIST）隶属于美国商务部，并在多个领域提供指导性指南。他们发布的《数字身份指南（SP 800-63）》中包含一系列与口令相关的主题。
- 英国国家网络安全中心（NCSC）属于网络安全领域的独立机构，同样在多个领域提供指导性指南，口令安全性自然也不例外。
- 开放Web应用程序安全项目（OWASP）是一个全球性非营利组织，致力于改善软件的安全性水平。他们在GitHub上提供的文档库整合了关于身份验证、口令存储等多个重要议题的备忘清单。
- Google就创建高强度口令、 提高Google账户安全性、口令管理、以及用户账户、授权以及口令管理的12项最佳实践等发布了相应的入门指南。

面向用户的口令注意事项

由于在线社区中的每一位成员随时都可能使用口令，因此从用户角度理解口令保护应该是讨论这项技术的理想起点。

创建口令

在创建口令时，您需要关注口令的长度与强度，同时也要考虑口令使用层面的安全问题。我们将在以下几节中具体探讨这些问题。

长度与强度

从最基础的层面出发，使用口令的目标在于防止未经授权的个人或者设备访问资源。这种预防的实现，一般依赖于在数学意义上难以猜测的用户口令，而猜测的具体难度被称为口令熵。简而言之，熵是一种面向复杂性与随机性的量化指标，而且以位（bit）为单位表示。

熵的计算

虽然目前行业对于如何计算口令的真实熵还存在一定争议，但以下公式能够给出较为简单的结果：*password length* × $\log_2$(*possible characters*) = *password entropy*
通过将可能字符概率的log2乘以口令字符长度，即可计算出口令熵的位值。口令的熵值越高，计算机就越难以通过暴力破解方式猜出、预测或者破解口令内容。在数学意义上，熵值每增加一位，口令的猜测难度都会成倍增加。例如，28位熵代表着2的28次方，或者268435456种口令组合可能性。全部由小写英文字母（共26个字母）组成，且长度为6个字符的口令，熵值约为28位。

从表一中，可以看到口令复杂度一路快速增长。右侧的词典列则表示从《牛津英语词典》中选出的完整单词（而非单一字符）。

Charset	a–z	a–z, A–Z	a–z, A–Z, 0–9	a–z, A–Z, 0–9, symbols	Full UTF-8	Dictionary
Charset size	**26**	**52**	**62**	**95**	**137,000**	**171,476 words**
Chars/ words	**Bits of entropy**					
4	19	23	24	26	68	70
6	28	34	36	39	102	104
8	38	46	48	53	137	139
12	56	68	71	79	205	209
16	75	91	95	105	273	278
32	150	182	191	210	546	556
60	282	342	357	394	1,024	1,043

Table 1.：口令复杂度的熵增情况

词典单词与口令短语

对于《牛津英语词典》这类常见的词汇工具书，其中包含有171476个词条（仅限于活词，不含理论存在但已不再使用的过时词汇），而这也代表着我们“字符”集的整体大小。如果从中选择一个单词，则相当于从一份包含171476个字符的字母表中选择了一个字符作为口令。使用前面提到的公式，使用单一词典建立的口令具有：1 x $\log_2(171,476)$ = 17位熵值

如果将口令的长度增加到4个单词（即创建包含4个单词的口令短语），则熵值将快速增长至70位。在数学意义上，除非使用量子计算机，否则该熵值已经足够强大、无法暴力破解。通过表二可以看到，即使每秒尝试1000种概率组合，这样的口令短语也可能需要长达270万年才能破解完成。

破解口令与彩虹表

由于口令通常以哈希形式存储在数据库内，因此黑客们当然希望掌握大量已知的口令哈希值。一旦黑客获得哈希后的口令，且其哈希值与已知口令的哈希值相匹配，攻击者就能还原出口令的真实内容。这种从口令到哈希值的映射集合被称为彩虹表（Rainbow Table），相当于口令哈希的一份逆向恢复目录。

彩虹表中的哈希值可通过多种方式生成。其中最基础的两种方法分别为：1）生成给定字符集的每一种具体可能字符组合；或者2）由以往被盗口令组成的列表。在后一种情况下，在获取口令列表时，我们需要对其中的每个条目进行哈希处理，并将结果存储在彩虹表中的明文口令旁边。当然，我们也可以使用完整的单词来生成口令短语彩虹表。

熟练的彩虹表制作者，能够运用统计数据、历史数据以及群体心理学制作出包含大量预计算哈希存储口令与口令短语的彩虹表。除了生成有效猜测这一基本目的之外，他们还经常会根据网站或者服务提供的口令复杂性规则定制自己的彩虹表。彩虹表制作者非常了解口令生成中的常见快捷方式，而且能够快速生成数量惊人的哈希，足以满足个人口令管理策略中的种种要求——例如“口令必须包含一个具有L337字符的词典收录词汇，外加一个特殊字符和一位数字”等。p455#w0rd9就完全符合这样的要求。

高级威胁者们的路子往往更野，他们会根据自己对目标的了解生成有针对性的彩虹表。他们能够经由网站、电子邮件、社交媒体内容、公开事实以及被盗信息生成自定义词典。如果用户手动选择口令短语中的单词，很可能会无意间选定自己熟悉或者偏好的字眼，并最终导致口令内容易被猜到。正如口令加盐（salted）部分所提到，更有效的方法是向各个单词加盐——这不仅降低了彩虹表的有效性，同时也能在体积更小的字符集之内显著降低可预测性（增加熵值）。

暴力破解与熵

了解了口令熵之后，接下来我们应当关注现代处理器的速度，了解计算机猜测特定口令内容所需要的时间。这通常被称为口令破解的“工作因子（work factor）”。例如，如果一台计算机每秒可以执行一百万次计算，而且口令内容存在1000万种可能的组合，则该计算机需要花费10秒以猜测口令的所有可能组合。

口令猜测率或者哈希率，在很大程度上取决于计算机算力以及使用的具体算法。生成口令的首选算法，包括Argon2、PBKDF2、Scrypt以及Brypt等，往往采用多种策略以进一步提升口令的暴力破解难度。例如，利用专用硬件预防数学计算加速、占用大量RAM的函数，甚至通过故意设计与哈希策略（例如MD5与SHA1）相冲突的推荐降低计算速度等。

在解决复杂性问题方面，熵是一种有趣且极具价值的方法，而其在评估计算机的实际口令猜测时长方面也非常实用。表二所示，为按破解时间检查口令复杂性。

Charset	a–z	a–z, A–Z	a–z, A–Z, 0–9	a–z, A–Z, 0–9, symbols	Full UTF-8	Dictionary
Charset size	**26**	**52**	**62**	**95**	**137,000**	**171,476 words**
Chars/ words	**Max time to crack @ 10,000,000 hash/sec**					
4	< 1 sec	< 1 sec	1.5 sec	8.1 sec	1.1 million years	2.7 million years
6	30.9 sec	33 min	1.6 hours	20.4 hours	21 trillion years	> universe lifespan
8	5.8 hours	2 months	8.3 months	21 years	> universe lifespan	> universe lifespan
12	302.6 years	1.2 million years	10 million years	1.7 billion years	> universe lifespan	> universe lifespan
16	138 million years	9 trillion years	151 trillion years	> universe lifespan	> universe lifespan	> universe lifespan
32	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan
60	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan	> universe lifespan

Table 2. 口令复杂度与破解时间

此表说明，如果计算机每秒能够执行1000万次猜测，则对于特定复杂度口令内容时，生成其中每一种可能组合所需要的时间。最右侧的词典一列代表从牛津英语词典中选择完整单词、而非使用单一字符。

可以看到，整个计算周期很快就超过了整个宇宙的预期生命周期（数百万亿年）。另外，使用6个随机字符生成的口令短语，在强度上大约与使用32个字符的纯小写字母口令基本相当。

根据以上解释，可以看到只有以下两个因素会提升熵值，从而增强口令的整体强度：

- 增加集合中的可用字符数量。
- 增加口令本身的长度。

通过在口令中使用更为丰富的字符集（具体包括大写字符、小写字符、数字、多语言普通字符、符号、表情符号等等），口令的计算机破解难度将快速提升。通过提升口令总长度，则可增加计算机猜测特定字符集内所有可能组合需要耗费的时长。

字符替换

很多人认为，可以使用外观相似的其他字符替代原本的单一字符，从而提高口令破解难度。这种处理方式被称为字符替换。例如，我们可以将单词“password”切换为“p@$$w0rd”的形式。这种替换方式比较常见，而且口令破解者也一般能够猜到替换方法。

最常见的替换方法几乎人尽皆知，因此即使替换也基本无法提高口令的实际复杂度。恶意攻击者拥有一套完整的口令哈希数据库，能够毫不费力地破解仅使用简单替换的口令内容。常用的彩虹表生成软件就具有这种常见字符替换功能，具体如下所示：

Character	Common replacements
A	4, /-\, /_\, @, /\, Д
B	8,|3, 13, |}, |:, |8, 18, 6, |B, |8, Io, |o, j3, ß
C	<, {, [, (, ©, ¢
D	), |}, |], |>
E	3, £, £, €
F	|=, ph, |#, |"
G	[, -, [+, 6, C-
H	4, |-|, [-], {-}, }-{, }{, |=|, [=], {=}, /-/, (-),)-(, :-:, I+I
I	1, |, !, 9
J	_|, _/, _7, _), _], _}
K	|<, 1<, I<, |{, I{
L	|_, |, 1,][
M	44, |V|, ^^, /\/\, /X\, []V][, []V[],][\V/][, (V),//., .\\, N\

N	||, /\/, /\/,][\\][, И
O	0, (), [], {}, <>, Ø, oh
P	|o, |O, |>, |*, |°, |D, /o, []D, |7
Q	O_, 9, (,), 0, kw
R	|2, 12, .-, |^, I2, Я, ®
S	5, $, §
T	7, +, 7`, '|' , `|` , ~|~ , -|-, ']['
U	|_|, _\, /_/, _/, (_), [_], {_}
V	\/
W	\/\/, (/\), \^/, |/|, \X/, \\', '//, VV, _|_/, \\/\\//, Ш, 2u, \V/
X	%, *, ><, }{,)(, Ж
Y	`/, ¥, |/, Ч
Z	2, 5, 7_, >_,(/)
0	O, D, space
1	I, L
2	Z, e
3	E, ❤
4	h, A
5	S
6	b, G
7	T, j
8	X
9	g, J

Table 3. 常见字符替换方法

这些方法的意义，在于通过增加口令随机性来提高熵值。从数学意义来看，更合理的方法是使用随机字符对口令进行加盐（salted）处理，具体细节将在后文中的口令加盐部分阐述。

曾用口令

在多个站点及应用程序上重复使用口令，会大大降低口令的保护能力。之所以会这样，是因为黑客通常会利用以往被盗用过的口令生成彩虹表。表内包含大量此前泄露或者已知的口令内容。黑客可以利用这份列表查找其他系统中的口令重复使用情况，即撞库。实际上，任何出现在词典文件或者彩虹表内的可用密钥及口令，在保护能力方面都与“password”这类低级错误一样不可靠。更合理的作法是做出“有罪推定”，即假设任何已经泄露的口令都会在一秒之内被现代口令破解软件所破解。

口令短语

口令中的每一个字符位，都存在着几十甚至上百个可能的选项；但口令短语会从词典中选出固定的字符组合，而且总单词数量达到十几万个。如果使用得当，口令短语足以建立起几乎坚不可摧的安全屏障。

Xkcd漫画向来以简单明了、直击核心而著称，我们也不妨就从这些有趣的画面入手，聊聊口令安全性问题。在示例中，作者通过易于记忆的方式选择四个常用单词组成了强口令。这种组合在必须亲自记忆口令且无法使用口令管理器时，往往能够收到奇效。举例来说，大家可以在零售商店的自助服务台或者其他独立的设备/系统上使用这类口令组合。

口令短语由随机选择的词典单词组成，因此口令短语本身的强度主要取决于词典内词汇量的大小与选择方式的随机性水平。请务必使用完全随机的词汇选择方式，例如从《牛津英语词典》中的171476个单词中随机选择——这种方法在安全性方面远远高于选择宠物的名字、所在街道或者您喜爱的乐队等等。这是因为后者所包含的单词总量往往只有几十个，这会把口令内容的复杂度从数万亿种组合减少至数百万种，最终令攻击者在几秒钟内即破解成功。

总而言之，完善口令短语的秘诀，在于真正以随机方式从大型词典中选取词汇。

口令加盐

您可以通过添加随机字符来增加口令乃至口令短语的复杂性。这种提升口令强度的方法，被称为加盐。加盐技术能够增加口令的总体随机性，同时也不致过度加大口令的记忆难度。在口令层面来看，加盐的本质是向口令内容中添加真正的随机性，从而抵御词典攻击并保证口令内容更加难以猜测。

这里以“DogCatFishRabbit”口令短语为例。这是一条非常合理的口令短语，其中包含16个大小写混合字符，能够产生91位熵值。但是，这条口令短语的问题在于选词随机性过低。为了解决问题，我们可以注入一些随机字符来提高口令短语的整体强度。当使用相同的口令内容，并选择字符“4”、“$”、“{”以及“z”来加盐，我们就能最终得到“D4ogCa$tFis{hRabzbit”。这是一条典型的混合口令/口令短语，长度为20个字符（131位熵值），且不会受到口令彩虹表攻击的影响。

创建强口令

要创建出高强度口令，在采用随机性保障策略的同时，大家也应避免使用易被猜测的模式。强口令需要满足以下规则：

- 避免使用众所周知的字符替换方法。
- 尽可能扩大字符集规模。
- 使用足以抵御现代黑客技术的口令长度。
- 不要使用日常生活中常用的数字、单词或者短语。

这些规则的存在，导致用户很难自主创造出符合要求的强口令。一串很长且包含完全随机字符的口令虽然最难猜测，但由于内容太过反直觉，也导致用户几乎不可能记住。

一种可行的方法是使用口令管理器。如果必须考虑使用强口令或者口令短语，我们建议您使用随机策略组合增加口令的熵。例如，如果口令的基础内容为“mango2”，那么以下几种技术能够显著提升口令的猜测难度：

- 使用Shift调整口令字符大小写并重复一遍，一次添加Shift，一次不添加：“mango2MANGO@”。
- 如果不支持特殊字符或者大小写混合，则可将原始口令内容反转一次：“mango22ognam”。
- 直接把每个字符重复一次：“mmaannggoo22”。
- 将两条不同的口令直接串联：“mango2tater7”。
- 将两条不同的口令内容混合起来，能够有效抵御词典攻击：“mTaAnTgEoR2&”。
- 同时使用以上多种方法，将一条低熵口令插入至另一条高熵口令内：“maaangoo2ttater7&RETATT@OOGNAAAM”。

如果您在创建口令时应用到其中几项规则，那么即使攻击者发现了口令的某些特征或者用于生成口令的部分字词，仍然很难完整还原您的口令内容。

口令管理器

口令管理器是现代安全中的重要组成部分。口令管理器相当于各类凭证与保密信息（例如信用卡号）的仓库，而且通常会配备相关软件以简化新口令创建、登录表单填写等流程。口令管理器使用一条主口令进行锁定，而且一般会采用多因素验证机制。

目前人们对口令管理器的普遍意见在于，其本质相当于把所有保密信息都存放在同一个高价值目标当中。一旦失去对该目标的控制，则意味着同时失去所有账户，由此带来的损失太过沉重。正因为如此，我们更应该在使用口令管理器时，选择那些可信度高、透明性好且具备多层安全控制机制的解决方案。

以下是判断一款口令管理器是否合格的基本要求：

- 能够加密静态数据（不仅限于口令）。
- 能够加密传输数据。
- 提供可撤销的恢复密钥，允许用户脱机输出并存储这些密钥。
- 支持除短信（文本消息）之外的多因素身份验证功能。
- 提供内置的口令生成器。

除此之外，我们还应根据实际需求关注口令管理器是否包含以下功能特性：

- 自托管。
- 口令管理器供应商无法恢复用户数据。
- 以允许用户自行编译的开源代码构建而成。
- 基于行为的安全控制机制。
- 跨平台支持能力。
- 自锁定数据库访问。
- 客户端管理的加密密钥。
- 不信任（TNO）安全原则。
  - 具有一定延迟性的部分或全部账户恢复功能，确保在用户死亡或发生其他紧急情况下移交所保存的敏感信息。

关于运营安全的其他常见问题

在使用新口令时，最好选择由口令管理器内置生成器提供的口令内容。需要强调的是，由免费在线网站生成的口令及口令虽然同样质量好、强度高，但生成的具体内容有可能被其他中间人、第三方脚本或者站点所窥探。万一出现这类状况，他人可能掌握您的IP地址，并通过您的浏览器收集到大量与身份信息相关的潜在口令列表。而且即使选择“生成多条口令、但只使用其中一条”这类迷惑性操作，我们仍无法显著降低口令内容外泄的风险。

下一步是确保建立备份计划。我们可能遭遇计算机盗窃、恶意黑客入侵、房屋火灾或者其他各类导致凭证丢失的紧急状况。因此，请在使用口令管理器的同时保留恢复代码的脱机副本，以便在口令管理器无法正常访问时及时找回账户凭证。

接下来，我们需要把思路拉回真实场景。每一位具有数字身份的用户，都应该考虑一旦自己发生人身意外，个人数据该如何处理。这里的个人数据

涵盖财务记录、电子邮件、加密货币乃至社交媒体。大多数服务不会在未接到法院命令的前提下，制定或者发布关于已故用户的数据处理策略。通过具有一定延迟性的部分或全部账户恢复功能，逝世者家属或其他关联人士能够在数周——而非数月甚至数年——之内继续访问数据。在功能设计方面，口令管理器应允许用户预先批准可以访问账户的个人，同时设置整个延迟周期的具体天数或周数。

最后，保障口令管理器的安全性，同时及时安装最新补丁修复程序。我们的整体安全水平，直接取决于最弱一环的实际质量。

安全提示问题……为什么始终难以完善

安全提示问题是账户安全中最薄弱的环节之一。服务运营商往往不会对这类问题的答案进行加密，且外部人士一般可以通过搜索公共记录、演绎法或者社会工程等途径发现这些答案。安全提示问题的实质，类似于一种后门口令。

解决这类问题的潜在方法之一，是利用专门针对安全提示问题的算法匹配与所提交问题并非直观关联的答案。下面来看一些与常见安全提示问题相关的

算法示例，可以看到这些方法都是在引导用户理解问题的真实含义，而非直接提问：

- **基础：第n字母**

安全提示问题算法的一项基本示例，就是直接输入问题内每个单词的第一个字符（也可以是其他特定位置的字符，统称为第n个字符）。单词之间不加标点符号、空格或者大写字母。例如，“What is your favorite kind of bird?（你最喜欢哪种鸟？）”就可以转换成“wiyfkob”形式。在第n字母算法当中，我们可以使用各个单词中的第一个、最后一个或者任何其他位置的字符，只要在各单词中坚持相同的位置选择即可。

- **中级：索引字母**

向算法中添加更复杂的加密内容，相当于对第n字母法的改进版本。例如，您可以选择一个自己牢记的电话号码，并将其作为选定字母的索引。例如，用电话号码“876-5309”代表“What is your favorite color?（你最喜欢的颜色是什么？）”如果将问题中的所有标点符号及空格删除，并从第一个字符开始计数，那么第8个代表问题中的“o”，第6个字母则是“s”。依此类推，最终得出的结果是“osyiaru”。

当然，大家不一定要止步于简单的数字形式。只要方法明确而且易于记忆，我们可以随意选择其他更灵活的映射与索引方式。

- **口令短语**

根据问题本身创建一条口令短语。例如，您可以输入安全提示问题中每个单词的最后一个字母，以此创建出一条易于记忆的短语：

"What was your high school mascot?（你就读高中的吉祥物是什么？）"

这个安全提示问题的末字母集合是“TSRHLT”。接下来，大家可以使用随机单词生成器并重复运行，直到将这些尾字母转换成新词汇的首字母，例如：

"tree surplus rebuilds halting loud title"。

类似于手动生成口令，用于生成安全提示问题答案的技术方案应该由您自己随意选择。需要注意的是，安全提示问题通常会对输入内容做出一定限制。有些系统禁止使用标点符号，有些则只允许使用英文字母，而且大部分不区分大小写。您需要结合使用策略尽可能提升安全级别。更重要的是，如果支持双因素身份验证功能，请务必用它替代简单的安全提示问题。如果不支持双因素验证，请考虑使用没有明确关联的口令短语或者随机生成的唯一字母字符串。

如果系统要求您输入自己的安全提示问题，请确保不要在问题中泄露任何可能导致账户入侵或者与其他账户相关的信息。

双因素验证

双因素身份验证（2FA）属于多因素身份验证（MFA）中的一类。后者的定义，是利用多种数据类型实现身份验证的安全保护策略。大家最熟悉的双因素验证方法，应该是由服务商通过短信服务向手机发送验证码。这一操作的核心，在于向网站证明当前操作者不仅清楚口令内容，同时也有权访问注册时绑定的手机。双因素验证分为多种类型，本文只简单介绍其中最常见的三种。

短信双因素验证存在一定隐患，而且极易受到现有入侵技术的影响。只要动机充分，攻击者完全可以劫持您的电话号码并拦截由该号码触发的双因素验证短信。以此为基础，他们能够全面侵入您的账户，甚至令电话号码无法正常使用。另外，短信消息本身未经加密，因此在安全水平上仅相当于电子邮件或者纸质便笺。

一次性口令（OTP）的本质，是创建一条在身份验证服务器与您所持有的加密令牌/移动应用间共享的密钥。二者会基于时间生成OTP码作为即时密钥。在建立共享后，双因素验证设备与服务器之间无需再次进行直接通信。只要双方的时钟保持同步，二者生成的OTP就会完全匹配。这种方法安全性较高，因为攻击者很难从设备中提取口令，而且OTP的生效周期只有几分钟，同时不依赖于短信等非安全通信渠道。

生物识别双因素验证在笔记本电脑和智能手机上较为常见，不同方案的可靠性与防欺诈能力也有所区别。目前，业界普遍认为常见的生物识别方案在安全性方面略优于短信验证码，但仍不及一次性口令。

各种双因素验证形式之间没有绝对的高下之分，其最核心的意义是在口令之外提供额外的身份验证因素。但在具体选择时（特别是选择短信验证等安全性较弱的选项时），请首先考量所保护资产的价值水平。

口令替代方案

不谈口令替代方案的指南，当然不能算是一份完整的口令应用指南。在各类媒体上，每隔几个月就会出现一篇宣称口令机制已经失效的文章。这类文章主要关注网络安全层面的细微差别，也有一些会着力宣传双因素验证或者口令管理器。

本节主要介绍以下几种可靠的口令替代方案。

数字证书

通过数字证书进行验证，可以算是目前最具历史的非口令保护机制之一。虽然个人用户接触不多，但数字证书被广泛应用在利用SSH进行远程服务器登录的场景当中。数字证书本身属于一种身份验证凭证，通常被存储在文本文件当中，支持用户使用口令进行加密。简而言之，数字证书代表的是存储在系统文件当中、且规模庞大到几乎不可能被猜出的口令内容。与其他保密信息一样，用户同样需要精心保护该系统文件。

SQRL

安全快速可靠登录协议（SQRL）是安全领域的最新成果，且专为最终用户在网站及应用程序上的身份验证场景而设计。SQRL用户需要在计算机或浏览器中运行小型客户端应用程序。该客户端不会直接提供站点所需要的口令，而是向用户交付可在目标移动应用或域名上进行身份验证的唯一公钥。

站点服务器将向客户端提供唯一值，客户端则使用本地私钥执行签名，并将该值发回。服务器利用公钥验证签名以证实用户身份。最重要的是，即使相关网站或者服务遭到入侵，攻击者也无法借此公开用户的登录凭证。

这套系统经过周密的设计考量与实施，既能够应对不断发展的在线威胁，也可解决实际使用者与安全控件之间的交互难题。在接下来的几年中，相信SQRL登录选项会逐步渗透进我们的日常生活。

生物识别

在理想条件下，生物识别可能是最为强大的身份验证技术之一。这种使用个人固有且唯一的属性实现身份验证的方法，也确实很有吸引力。但目前的大部分消费级生物识别安全系统（例如指纹、虹膜或者人脸识别）都较易受到欺诈攻击，因此难以被部署至高敏感度系统当中。此外，一旦您的生物特征数据泄露，还会引发其他严重后果。例如，当用户在餐厅的杯子上留下指纹时，或者如果攻击者获得了潜在目标的手部或面部高分辨率照片，生物识别机制将失去安全保护能力。最后，生物识别机制还难以找到有效的口令共享或者已故用户访问权交接方法。

好消息是，这一领域每年都在快速发展，而且生物识别技术非常适合用作双因素验证中的辅助型凭证。

基于设备的身份验证

这种越来越流行的身份验证机制，实际上就是利用一台设备上的活动会话对另一台设备上的用户进行身份验证。与双因素验证不同，这里把设备作为证明身份的主要手段。例如，当用户执行网站登录操作时，网站会向注册时留下的手机发送通知以进行验证。除了作为对传统登录的多因素验证补充之外，这种方法还适用于自助服务终端、控制台以及共享系统等难以或不宜输入口令的场景。基于设备的身份验证还可用于重新验证过期会话，或者帮助账户管理等高风险活动强化安全性。

扩展阅读

- 参阅关于用户账户、授权及口令管理的12种最佳实践。
- 参阅Google关于良好账户安全策略对劫持行为防御效果的研究。
- 参阅面向系统设计师的现代口令安全。
- 欢迎大家亲自体验其他Google Cloud Platform功能，此处为相关教程。