发现这样一个神奇的问题：老师或者教务处习惯使用弱密码，尤其是123。excel保护、虚拟机、板子登录都是。

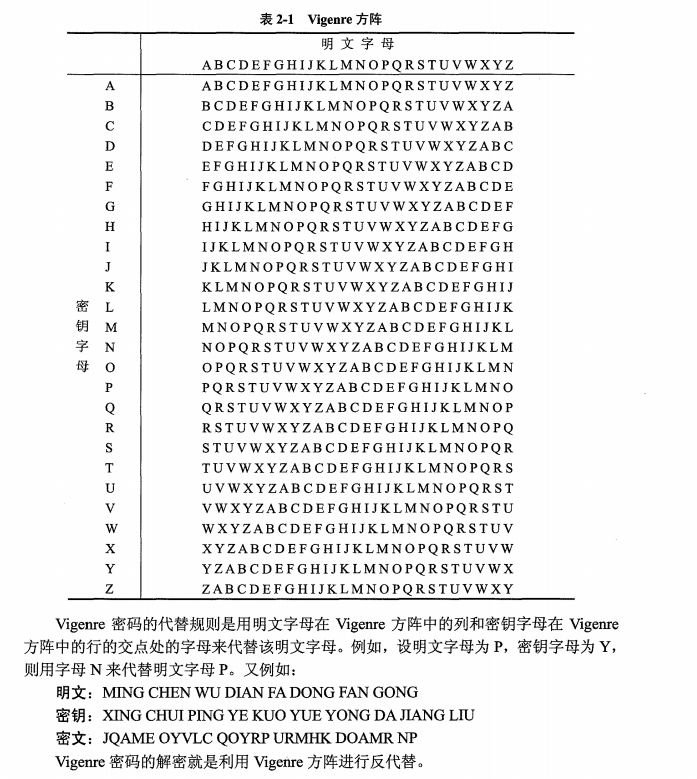
"一次一密"密码是理论上安全的密码，其余的密码都只能是计算上安全的密码。根据计算复杂性理论的研究， NP 类问题是困难的。 NPC 类问题是 NP 类中最难计算的一类问题。公钥密码的构造往往基于一个 NPC 问题，以此期望密码是计算上安全的。

在密码学中Nonce是一个只被使用一次的任意或非重复的随机数值。用于防止重放攻击。

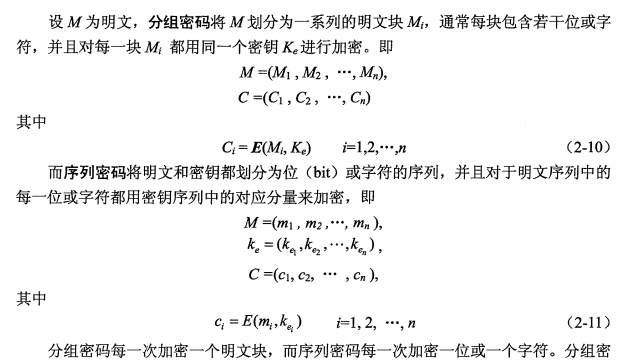
密码杂凑函数（Cryptographic hash function），又称为密码散列函数、加密散列函数，散列函数的一种。 杂凑函数是一种单向函数。

置换密码：写成方阵的形式（不存在的地方使用“空字符”替代），按照位置的排列从而得到一个新的密文。

代替密码：两个预定义字母表之间进行映射=》加法密码，乘法密码（j=ik mod n。对于明文字母表为英文字母表的情况 ， k 只能取 3，5, 7, 9, 11, 15, 17, 19, 21, 23 , 25 共 11 种不同的取值，k的取值与26没有公因数，比加法密码弱得多），仿射密码（j=ik1+ko mod n，这种线性函数将i映射到j上面），多表代替密码（典型：Vigenre 密码）

，代数密码（异或运算）

凯撒加密是指对字母进行变换，但是CTF可能并不拘泥于这个，没准就是简单地找ASCII规律。



LM哈希长度小于14位的，就是找死。

目前，我国许可并大力推广应用的通用密码算法是 SM2 椭圆曲线公钥密码算法、 SM3 密码杂凑算法和 SM4分组密码算法。同时，在智能电网、居民健康卡、社会保障等领域允许使用 SMl 分组密码算法

新型密码：生物密码，量子密码

公钥密码系统的加密算法ECC（椭圆曲线密码学）、RSA。

DH算法在设计时限于密钥交换的用途

硬件加速：使用CRC-32C的优势是INTEL处理器发布的SSE4.2的支持

**密码分析者攻击密码的方法主要有以下三种。**

**(1)穷举攻击。所谓穷举攻击是指，密码分析者采用依次试遍所有可能的密钥对所**

**获得的密文进行解密，直至得到正确的明文:或者用一个确定的密钥对所有可能的明文**

**进行加密，直至得到所获得的密文。显然，理论上，对于任何实用密码只要有足够的资**

**源，都可以用穷举攻击将其攻破。从平均角度讲，采用穷举攻击破译一个密码必须试遍**

**所有可能密钥的一半。**

**值得注意的是，如果分析者不得不采用穷举攻击时，他们往往会首先尝试那些可能**

**性最大的密钥。例如，基于用户为了容易记忆往往会选择一些短的数据或有意义的数据**

**作为口令(如姓名、生日、电话号码、邮件地址等)这样的事实，黑客在攻击用户口令**

**时往往首先尝试这些短的和有意义的口令。这一事实告诉我们，为了口令的安全用户不**

**应选择这种短的数据或有意义的数据作为口令。**

**(2) 数学分析攻击。所谓数学分析攻击是指密码分析者针对加解密算法的数学基础**

**和某些密码学特性，通过数学求解的方法来破译密码。数学分析攻击是对基于数学难题**

**的各种密码的主要威盼。为了对抗这种数学分析攻击，应当选用具有坚实数学基础和足**

**够复杂的加解密算法。**

**其中最神奇的就是分析字母表中一个字母，两个字母，三个字母等等范围内出现的概率。**

**(3)基于物理的攻击。侧信道密码分析利用密码系统实现时泄露的额外信息，推导**

**密码系统中的秘密参数。主要方法包括功耗攻击、电磁场攻击和时间攻击等。其中功耗**

**攻击是最常用的手段之一，包括简单功耗分析攻击 (SimplePower Analysis attacks, SPA)和差分功耗分析攻击 (Differential Power Analysis attacks，DPA) ，与传统密码分析学相比，这些攻击手鼓攻击效果显著。有效性远高于基于数学进行密码分析的方法，因此给密码设备带来了严重的威胁。**

差分攻击是通过比较分析有特定区别的明文在通过加密后的变化传播情况来攻击密码算法的。

目前可用于密码破译的量子计算算法主要有Grover 算法和 Shor 算法。对于密码破译来说， Grover 算法的作用相当于把密码的密钥长度减少一半。而 Shor 算法则可以对目前广泛使用的 RSA、 EIGamal、 ECC 公钥密码和 DH 密钥协商协议进行有效攻击。但是目前加拿大的量子计算机属于专用型量子计算机，它能够执行 Grover算法，尚不能执行 Shor 算法。美国加州大学圣芭芭拉分校的量子计算机可以执行 Shor算法，但量子位数太少。

如果你在过去的十二年里不是生活在原始森林的话，那么你一定知道密码学哈希函数SHA-1是存在问题的，简而言之，SHA-1没有我们想象中的那么安全。SHA-1会生成160位的摘要，这意味着我们需要进行大约2^80次操作才能出现一次哈希碰撞（多亏了Birthday攻击）。但是在2005年密码学专家通过研究发现，发生一次哈希碰撞其实并不需要这么多次的操作，大概只需要2^65次操作就可以实现了。这就非常糟糕了，如果现实生活中可以实现哈希碰撞的话，那么这就会让很多使用SHA-1的应用程序变得更加的不安全。而这个操作次数，即2^65次，是一个比较接近实际情况的数字。如果你想用自己的笔记本电脑来进行计算的话，可能需要好几年的时间，但是对于那些手头上有资金有技术的大型组织而言，计算出一次哈希碰撞可能只需要一两周的时间。

BLAKE2有两大主要版本：BLAKE2b和BLAKE2s。BLAKE2b是BLAKE的64位版本，它可以生成最高512位的任意长度哈希。BLAKE2s是BLAKE的32位版本，它可以生成最高256位的任意长度哈希

SHA-2(其中包括SHA-256、SHA-384和SHA-512)哈希函数可以看作是安全系数更高的SHA-1，而SHA-2目前还没有被破解。但是跟BLAKE2相比，SHA-2哈希函数在性能和速度方面明显要处于劣势。

SHA-1碰撞在文件方面危害？比如证书被破解。

彩虹表是预先计算好的密码散列列表。直接查数据进行匹配，少了密码生成与hash转换，非常快。我们可以先生成彩虹表。

"散列链"（彩虹链）是为了降低传统做法空间要求的技术。我们的想法是定义一个衰减函数 R 把散列值变换成另一字符串。通过交替运算H函数和R函数，形成交替的密码和散列值链条。

例如:假设密码是6个小写字母，散列值为32位长，链条看起来可能是这样的：

aaaaaa ---> 281DAF40 ---> sgfnyd ---> 920ECF10 ---> kiebgt

\_\_\_\_\_\_\_H()\_\_\_\_\_\_\_\_\_\_\_R()\_\_\_\_\_\_\_\_\_H()\_\_\_\_\_\_\_\_\_\_\_R()\_\_\_\_\_\_\_\_

要生成一个表，我们选择一组随机的初始密码，每一个密码计算一个固定长度K的链，并只存储每一个链的第一个和最后一个密码。第一密码被称为始点，最后一个被称为末点。在上面例举的链中，“aaaaaa”就是始点，“kiebgt”就是末点，其他密码（或散列值）并不被保存。

假如给定一个散列值h ，我们要反运算（找到对应的密码），计算出一个链，以对h应用R开始，然后H，然后R，一直继续。如果在该运算过程中的任何点（每次应用R后），我们发现该点的值的匹配我们生成的表中的一个末点，那么我们就得到了相应的始点，用这个始点来重新计算链。这条链会有不错的几率包含值h，而如果确实包含，链中h前面紧接的值就是我们所寻求的密码p。

md5和SHA-1已经可以被家用PC破解了，关键人物：王小云，似乎说不能说是破解，而应该是碰撞。

echo '4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbfa202a8284bf36e8e4b55b35f427593d849676da0d1d55d8360fb5f07fea2' | xxd -r -p | tee >/dev/null >(md5sum) >(sha1sum) #>(必须要是连着的

echo '4dc968ff0ee35c209572d4777b721587d36fa7b21bdc56b74a3dc0783e7b9518afbfa200a8284bf36e8e4b55b35f427593d849676da0d1555d8360fb5f07fea2' | xxd -r -p | tee >/dev/null >(md5sum) >(sha1sum)

这两字符串md5计算是一样的，sha计算不同。

还有的是文件会这样，那你就可以将文件内容读出来啊，二进制读取，如果要web发送给服务器，比如post，那就进行urlcode编码就好。不是二进制读出然后显示出来是乱码格式那就最好不要直接这么做拿来用！不过理论上应该是可以做到的，不知道是不是终端显示了以后二进制也会变？？？？

基于概率的口令猜测：适合已知一部分用来猜测剩下一部分，本质上感觉还是字典破解

（1）马尔可夫链又叫做离散时间马尔可夫链，是在状态空间内从一个状态转换到另一状

态的随机过程。P(password)=P(p)P(alp)P(slpa)P(slpas). . .P(dlpasswor)，如果变种P(password)=P(p)P(alp)P(slpa)P(slpas). . .P(dlswor)。

（2）基于 PCFG 方法

http://cmd5.com 彩虹表

md5 标准md5，32位或16位

md5(md5($pass)) b80c9c5f86de74f0090fc1a88b27ef34 第一次加密后，结果转换成小写，对结果再加密一次

md5(md5(md5($pass))) e57941ff9000aedb44eb2fa13f6e3e3c 第一次加密后，结果转换成小写，对结果再加密一次,结果转换成小写，对结果再加密一次

MD5(MD5($pass)) bb7ff6177ee612ef9dc6acd3a9ea7ea9 第一次加密后，结果转换成大写，对结果再加密一次

MD5(MD5(MD5($pass))) 36d627bd562e83ab995fb1fdf59c95d9 第一次加密后，结果转换成大写，对结果再加密一次,结果转换成大写，对结果再加密一次

sha1 f03e8a370aa8dc80f63a6d67401a692ae72fa530 密文长度必须为40位

md4 c0a27f801162b8b862cd5f5a1a66e85a 32位

mysql 29596332026fd206 老MYSQL数据库用的，16位，且第1位和第7位必须为0-8

mysql5 b34c662f720236babfc1b3f75203a80e1009844a 新版本MySql数据库用的

md5($pass.$salt) 9393dc56f0c683b7bba9b3751d0f6a46:OTD6v4c8I3Zid2AL 在密码后附加一个字符串再加密。

md5($salt.$pass) 5610604c157ef1d0fb33911542e5b06f:zg

5610604c157ef1d0fb33911542e5b06f zg 在密码前附加一个字符串再加密。

Md5(Phpbb3) $H$912345678Mw/BjmincvnSS94/STawW/ Linux

Md5(Wordpress) $P$B12345678/c7bOMfLdQB9B/ypks8iB/ Linux

Md5(Unix) $1$12345678$kbapHduhihjieYIUP66Xt/ Linux

Des(Unix) af.kPXROLU9uY Linux（猜测特征：比较短）

hash中的salt表示在转换结束后加入hash

MD4（Message Digest v4）基于32位操作数来实现的。MD5和SHA1都是基于MD4。MD5是按照512位分组进行输入。SHA1对于程度小于264的输入，都会产生长度为20Bytes的散列值。

证书：

银行在浏览器上的证书，通常就是ID+口令

在域环境中，Kerberos协议被用来作身份认证，上图所示即为一次简单的身份认证流程，具体细节可以参考相关资料，这里仅介绍几个名词：

KDC(Key Distribution Center)： 密钥分发中心，里面包含两个服务：AS和TGS

AS(Authentication Server)： 身份认证服务

TGS(Ticket Granting Server)： 票据授予服务

TGT(Ticket Granting Ticket): 由身份认证服务授予的票据，用于身份认证，存储在内存，默认有效期为10小时

Pass The Ticket： 如果我们能够拿到用户的TGT，并将其导入到内存，就可以冒充该用户获得其访问权限

每个用户的Ticket都是由krbtgt的密码Hash来生成的

pass-the-hash: pass基于NTLM哈希的哈希

pass-the-ticket：pass基于Kerberos票据的票据.最常见的手段就是使用黄金票据（Golden Ticket）和银票据（Silver Ticket）.在我们成功获得域控权限后，就可以导出域控内存中的Ticket，在默认的10个小时以内都可以利用来登录域控，通过mimikatz导出DC内存中的Ticket，sekurlsa::tickets /export，然后在普通主机中导入：mimikatz "kerberos::ptt C:\test\[0;2d87a]-2-0-40e00000-a@krbtgt-TEST.LOCAL.kirbi"。

Golden Ticket：伪造的TGT(Ticket Granting Ticket)，可以用于任何服务,在DC上通过mimikatz log "lsadump::dcsync /domain:test.local /user:krbtgt" 导出krbtgt的hash，接着配合导出的hahs构造一个golden ticket：mimikatz "kerberos::golden /domain:xx /sid:S-1-5-xxxx /aes256:xxxx /user:god /ticket:gold.kirbi",生成Golden Ticket不仅可以使用aes256，也可用krbtgt的NTLM hash。需要同域控通信。

Silver Ticket:伪造的TGS(Ticket Granting Server)ticket,权限是有限的额只能访问指定的服务权限。Silver Ticket是由服务账户（通常为计算机账户）hash加密，gloden则是通过krbtgt的密码Hash来生成的。Silver Ticket在使用的过程不需要同域控通信。通过

mimikatz log "sekurlsa::logonpasswords" 导出本地账户的hash，然后在域主机中

mimikatz "kerberos::silver /domain:xxx /sid:S-1-5-xxxxxx /target:WIN-8VVLRPIAJB0.test.local /service:cifs /rc4:xxxx /user:silver /ptt" 创建出一个silver ticket。

/domain

/sid

/target:目标服务器的域名全称，此处为域控的全称

/service：目标服务器上面的kerberos服务，此处为cifs

/rc4：计算机账户的NTLM hash，域控主机的计算机账户

/user：要伪造的用户名，此处可用silver测试

散列注入软件：

windows：

wce win7中需要管理员权限，2003中直接使用

Pass the hash (NTLM)

Pass the ticket (Kerberos)

‘Steal’ Credentials from memory 当设计密码的验证（用户登录以及相关的windowsAPI，设计指定用户的windows service开启， 执行了runas /u:guest cmd.exe命令），就会出现在内存中。WCE is the first and only tool that can read NTLM credentials stored by Windows in memory without injecting code。WCE is able to locate and understand the undocumented structures used by Windows to store the credentials, find encryption keys and decrypt credentials just by reading the system's memory. 失败了以后就会进行代码注入从而获取内存中的NTLM，可以使用-f参数取消第二步。通过SMB shares等，可以远程使用steal。

-f 强制仅仅使用非注入的内存NTLM获取技术

wce.exe -g <cleartext password> 结合当前登录的用户计算NTML，然后就可以用于-s

wce.exe -s testuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537 -c cmd.exe 更改NTLM的同时，创建并使用了新logon session产生的。

散列爆破软件：

windows：

Hash Suite Pro

L0phtCracker7

Ophcrack

Cain(非常适合用来破解LM、NTLM、Kerberos，有很多种破解的方法)

linux：

John the Ripper password cracker

crunch 密码字典生成工具

OclHashcat(用了CUDA，Compute Unified Device Architecture是一种由NVIDIA推出的通用并行计算架构)

获取Hash：pwndump、GetHashes

windows散列文件：

windows的散列保存在SAM（安全账号管理器）文件里面，在NT内核的windows系统中，包括windows 2000及后续版本，都在

C:\Windows\System32\config\SAM，不能打开这个文件说“文件已经在System”中打开，所以也就不能复制。这是因为系统启动以后SAM文件会被锁定，同时它也会被加密。

绕过锁定的方式：双系统。

绕过加密的方式：可以通过Samdump2等dump来提取出散列，书上说是利用C:\Windows\System32\config\SYSTEM来解密:

samdump2 SYSTEM SAM > /tmp/hash.txt

有一些系统里面就会需要加一步通过bkhive来从系统配置单元（Hive）中提取Syskey启动密钥（Bootkey）：

bkhive SYSTEM bootkey.txt

samdump2 SAM bootkey.txt > /tmp/hash.txt

Meterpreter的hashdump会直接绕过windows的安全机制并解密文件来获取hash。

散列攻击：

密码重置的话就是直接覆盖SAM文件，kali live提供的工具就是chntpw（-i来启动交互模式）:

chntpw -i /mnt/xxx/.../SAM（-i 与程序交互）

对散列的破解都是碰撞，因为散列从技术上实现了不可逆。每秒破解次数c/s。

如果散列是在本地计算并发送的，那么就可以重放攻击。

散列漏洞：

原本微软使用的散列技术时Lan Manager(LM)：字符限定为14字符（不足就要null值来填充1，过多就截断）-》密码分成7、7两段-》密码串大写-》散列函数。前面几步降低了破解的时间。LM（IBM创造）进一步就是NTLM（微软改进）。

hash传递攻击：

可以直接使用用户的hash来作为密码进行登录。很多windows协议都需要用户提供hash值进行验证，而不是密码。所以一旦电脑被攻破就能dump hash并作为自己的登录使用。目前见过的有msf中smb/psexec。

linux散列文件：

/etc/shadow但是查看可能需要权限

/etc/passwd默认是可以被所有用户读取

crunch：一般密码应该随身携带，使用这个程序的时候说明是不得已

/usr/share/crunch/charset.lst 这个文件里面存放的是<charset string>的配置，比如：

hex-lower = [0123456789abcdef]

crunch <min-len> <max-len> [<charset string>] [options]

crunch 1 6 abcdefg\ #生成 最小为1,最大为6.由abcdefg和空格为元素的所有组合（/代表空格）

crunch 1 8 -f charset.lst mixalpha-numeric-all-space -o wordlist.txt #mixalpha-numeric-all-space项目包含最常见的元素组合（即大小写字母+数字+常见符号）

绕过权限获取/etc/shadow：通过unshadow命令（如果有这的话）来合并/etc/shadow和/etc/passwd

unshadow /etc/passwd /etc/shadow > /tmp/hash.txt 这是John里面自带的程序

破解之前记得清理掉无用的账户

下面不关于密码学，与windows的实现有关：

在WinlogonHack之前有一个Gina木马主要用来截取Windows 2000下的密码，WinlogonHack主要用于截取Windows XP以及Windows 2003 Server。安装以后只要有RPC登入就会记录密码到一个文件中。执行"Uninstall.bat"即可自动卸载该程序（关键在于恢复被改动的dll）。原理：

Gina.dll在NT/2000中提供登录支持，后来再XP及后续版本中改为Msgina.dll（正常情况下在XP下是945KB，在2003下是1153KB）。文件改不好了的后果就是不能RPC登录。WinlogonHack Hook住dll中的WlxLoggedOutSAS函数，从而记录密码。WinLogon初始化的时候会创建3个界面：winlogon桌面（C-A-Delete等这些），应用程序桌面（有“此电脑”的这个桌面），屏幕保护桌面

serv-u生成密码hash的方式：

1.随机产生两个字符

2.这两个字符加上你的密码进行md5

3.在产生的MD5大写结果前加上第一步随机产生的两个字符

密码长度增加了两位？不，其实没有什么变化....开头两位都知道了，那么对爆破而言工作量差不多，只不过彩虹表方法被影响了

配置文件在安装目录下的ServUDaemon.ini，在配置文件的[GLOBAL]下加上ReloadSettings=True，这样Serve-U就会自动刷新配置文件，然后去掉这个参数。配置文件中还有对上传与下载速度、最大连接数的限制。

1.绕过，不用密码来解决问题

2.一次爆破的基本密码时：社工密码加上弱密码（我是指常用密码）；尽力爆破；hash碰撞

3.覆盖密码文件从而修改

凡是加密的文件比如说Access可以被加密，就会有对应的解密程序。

京东已经实现对密码的加密，但是在更改密码的时候依旧提供对密码相近程度的判断，一般提供这种判断就是明文存储吗，因为需要计算两个字符串之间的距离（信息检索里面已经介绍过了）所以京东就是明文存储或者是可逆的密码存储。除非将这个密码相近程度判断的功能做的不怎么样，计算多个距离为多少的密码，通过正向计算得到不可逆的对应的hash，如果匹配上级就会给提示，但是对于长密码，字符串步数每一次增加都是快速增长的资源消耗。正向计算的话，只有几次还是很简单的，一般的网站都在使用。