Outlook passwords

Passcape Software

1 介绍

"Use strong passwords that combine upper- and lowercase letters, numbers, and symbols. Weak passwords don't mix these elements. Strong password: Y6dh!et5. Weak password: House27."

Excerpt from MS Office Outlook user manual

This article was originally meant to tell you about a funny passwords collisions in Outlook's **PST** files. Later on, it was expanded to demonstrate that despite the drawbacks, Outlook's advantages far exceed its closest competitors, as well as to explain the techniques used for storing personal data. Besides, it is very convenient to follow the development of the cryptography using Outlook as an example. It can be generally projected to the development of the entire line of Windows operating system as a whole.

2 PST保护密码

因此，让我们开始与点，微软Office Outlook的.PST文件是文件类型的数据存储在本地计算机上，它存储的联系人，笔记，电子邮件消息，和其他项目按一定顺序排列。一个.PST文件可以作为传递电子邮件信息的默认位置。它也可以用于排序和备份数据。

为了保护PST文件的内容，并限制第三方未经授权的访问，人们可以设置一个长达15个字符的密码（图1）。在这种情况下，人们普遍认为，PST文件不能被打开，除非知道原始密码。让我们来看看这是多么真实。

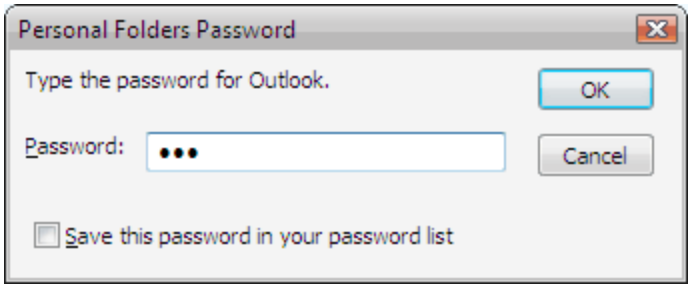

图1. Outlook PST密码对话框。

PST文件的访问密码既不会被记住，也不会以明确的形式存储。相反，计算机计算密码的哈希值，并将其存储在.PST文件中，或者，如果选择 "将此密码保存在您的密码列表"（图1），在Windows注册表，另外加密。

最有趣的是，密码散列计算算法不是真正的散列算法 - 它是一个简单的CRC32校验计算程序。CRC32是一种冗余检查算法，它肯定不是一个散列算法。由于这样或那样的原因，微软决定使用这种算法，而不是像SHA-1这样更强大的算法。作为对旧版本Outlook的继承，微软已经很久没有改变这个算法了，似乎是被向后兼容的想法所引导。另一方面，我们仍然不清楚，为什么在Outlook 2003发布时没有改变（Outlook 2003的.PST格式有64位的内部地址，与以前的版本不兼容）。

因此，密码hash似乎是其校验和的32位。让我们仔细看看CRC32的执行情况：

DWORD CPstReader::Crc32(LPBYTE pPassword)

```
{
    assert( pPassword );

    //set initial crc to zero
    DWORD crc=0;

    //till the end of string
    while ( *pPassword )
        crc = (crc>>8) ^pCRCTable[(BYTE)crc ^(*pPassword++)];

    return crc;
}
```

从源代码的片段可以看出，密码（pPassword）被输入，其校验和出现在输出端。CRC32最脆弱的地方是，密码哈希值的32位长度显然不够长，因此两个不同的密码的校验和可能会匹配！例如，密码1和密码2的校验和是一样的。例如，密码1和orxgnm或者mozart和2920347097的校验和是一样的。如果你认为这种碰撞很罕见，那你就大错特错了。

这些碰撞是多么令人惊讶的事情啊　相反的依赖性也经常发生：密码越长，该密码的碰撞匹配就越简单。让我们来看看Outlook用户手册中的例子，其中的摘录被选作本文的序言，这是特意选择的。有一个更简单的组合，可能很容易取代我们的　"可靠　"密码Y6dh！et5　-　这是一个5字符的字符串JlSfw。

然而，另一个有趣的观察：如果存储在.PST文件中的校验和等于零，程序会 "认为 "没有设置密码。然而，由于我们知道具有相同校验和的密码确实存在，我们可以假设存在密码，其校验和也等于零。这样的密码确实存在。下面是这个大列表中的一小部分。1Rj78C, 5J8j84, ArTniW.

Crc32("1Rj78C")=0，Crc32("5J8j84")=0，Crc32("ArTniW")=0。如果我们设置这样一个密码来保护我们的.PST文件，实际上我们的文件是没有保护的，每当有人试图在下一次访问它时，它甚至不会提示输入密码。不相信吗？- 自己试试吧。

一项实验表明，使用暴力攻击恢复Outlook哈希密码平均需要约一分钟。然而，对CRC32的加密分析表明，该算法对短密码（最多4个字符）是完全可逆的，对所有其他密码是部分可逆的。这意味着，人们几乎可以立即恢复原始密码或其CRC32等效密码，这对Outlook来说将是不可区分的。事实证明，它需要不超过7个字符来选择一个碰撞（密码与原始密码的校验和相同）。

3 PST加密

如果我们看一下PST文件选项（图2），我们可能会注意到，Outlook，除了其他一切，允许加密的内容。在这种情况下，密码校验和不会以开放的形式保留在PST文件中，相反，它还会用加密算法进行额外的加密。让我们回顾一下用于加密的算法。

在创建一个新的.PST文件时，Outlook提示我们在3种文件类型中进行选择：

1. 不加密
2. 可压缩的加密
3. 强加密

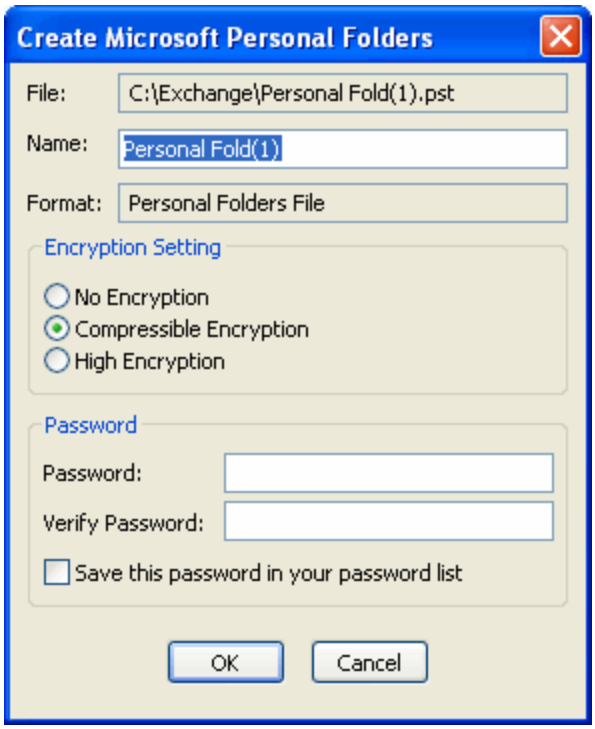

图2. 创建新的PST文件。

如果没有为PST文件选择加密，用户的所有数据：联系人、信息、密码等都将以开放的形式存储，供其他用户使用。这些数据可以用例如文本编辑程序来查看。

可压缩加密算法的方式是，每一个被加密的字符都被从一个特殊的表格中取出的不同字符所替代。以下是该算法：

```
BOOL CPstReader::Decrypt1( LPBYTE buf, int iSize )
{
   assert( buf );

   BYTE y=0;
   int x=0;

   //Check input buffer
   if ( buf==NULL )
      return FALSE;

   //Check encryption type
   if ( m_pst.encryption!=PST_ENCRYPT_COMPESSIBLE )
      return FALSE;

   //actual decryption
   while ( iSize-- )
   {
      y=buf[x];
      buf[x++]=m_pTable[y];
   }

   return TRUE;
```

```
}
```

替换表(m_pTable)的创建是有意义的，它允许进一步压缩用该算法加密的文本的最佳方式。然而，该算法本身并没有压缩内容；它只是为此创造了有利的条件。

强加密也是第一种替换算法的一种。然而，与前一种不同的是，它提供了更强大的加密功能。这种算法的另一个区别是，用这种算法加密的PST文件不能像用第一种方法那样紧凑

```
BOOL CPstReader::Decrypt2( LPBYTE buf, int iSize, DWORD id )
{
    assert( buf );

    int x=0;
    BYTE y=0;
    WORD wSalt;

    //Check input buffer
    if ( buf==NULL )
        return FALSE;

    //Check encryption type
    if ( m_pst.encryption!=PST_ENCRYPT_STRONG )
        return FALSE;

    //prepare encryption key from block ID
    wSalt=HIWORD(id) ^LOWORD(id);

    //actual decryption
    while ( iSize-- )
    {
        y=buf[x];

        y+=LOBYTE(wSalt);
        y=m_pTable2[y];

        y+=HIBYTE(wSalt);
        y=m_pTable2[y+0x100];

        y-=HIBYTE(wSalt);
        y=m_pTable2[y+0x200];

        buf[x++]= y - LOBYTE(wSalt++);
    }

    return TRUE;
}
```

要恢复一个数据块，必须知道该数据块的标识符。如果没有这个标识符，恢复过程就会显得有些困难。同样，如果我们仔细看一下该算法的源代码，我们可以很容易地注意到，只使用了一个16位的块标识符，这使得用暴力攻击挑选加密密钥的迭代次数大大减少。

4 储存电子邮件账户密码的技术

当你阅读这篇文章时，你可能会得到一个有点错误的想法，即MS Outlook只使用弱的、不可靠的密码加密算法。这并不完全是这样。俗话说，在比较中会发现真相。更重要的是，我们只分析了PST。现在让我们回顾和比较一下Outlook用于电子邮件帐户的密码加密机制与其他流行的程序。

大多数流行的电子邮件客户端根本不关心用户存储在程序中的密码的安全。例如，Eudora、TheBat！或Netscape的旧版本使用古老的BASE64算法或其衍生算法来加密用户的数据。IncrediMail只是简单地将XOR伽玛应用于原始密码（不愿意称这个东西为加密），老版本的Opera浏览器根本没有对电子邮件密码进行加密；它们将密码存储为纯文本！。

这个问题在流行的电子邮件客户端Thunderbird、Opera M2和Outlook Express的新版本中要好一点。它们都使用可靠的、经过时间验证的算法，并带有主控键。这通常是MD5+RC4或SHA+3DES或其衍生物的组合。雷鸟、Opera M2和Outlook Express将它们的主密钥和加密密钥与加密的密码一起存储。这允许无缝（或几乎无缝）地恢复这些密码。

关于Outlook（顺便说一下，不要把它和Outlook Express混为一谈），在这里我们看到了最耐人寻味的画面。Outlook的电子邮件账户的密码存储技术的整个发展年表可以分为四个时期

- 史前时代
- 石器时代
- 中世纪
- 技术进步时代

现在有了更多细节。

4.1 史前时代

Prehistoric Age - this is the period of the first steps. They say the first versions of the program were capable of encrypting passwords stored in the registry with the **BASE64** algorithm. That already was an achievement by that time's standards. To recover such passwords, one needed a calculator and a couple of convolutions in the brain.

4.2 石器时代

Stone Age - Outlook 9x - the time of first experiments. A new encryption algorithm that uses the encryption key and unique record identifier appears at this time. It's the age's know-how. Data encrypted with this algorithm could not be recovered without the key and record identifier. Many of today's e-mail clients cannot brag about having such algorithm in their

arsenal even today. However, the general idea of that type of encryption was darkened by one crucial weakness - the encryption key and record identifier were stored in the registry, along with the encrypted data.

4.3 中世纪

Middle Ages - Outlook 2000 - the first standards. E-mail account passwords were now kept in Windows' Protected Storage (one of the upcoming articles will cover the Protected Storage in greater detail). Here is the password decryption algorithm in Protected Storage:

1. Key1 is created for decrypting master key, using SHA(Salt) + SHA(SID) + SHA(Salt). Salt is a global constant. SID is user identifier.
2. Key2 is created for decrypting master key, using SHA(MKSalt) + SHA(key1). MKSalt is binary data unique for each master key stored with it. Key1 is the 20 bytes of data received on the previous step.
3. Master key is decrypted with the DES algorithm and Key2.
4. The recovered master key participates in the decryption of the data encryption key. The data encryption key is stored with data itself and is different for each data record. It consists of 16 bytes, the first half of which is used for the decryption, and the second half is used for the validity check.
5. Now, using this decrypted data key one can decrypt the data record itself (passwords, credentials and other sensitive information).

What a great and stylish idea! This algorithm's major features:

- Single master key for all records. Therefore, it's enough to decrypt it once (steps 1-3). This speeds up the decryption process without bringing down the security level in whole.
- User SID participates in the decryption of the master key; therefore, each user will have a unique master key. Thus, it becomes clear that it is impossible to decrypt the data unless one knows the SID (which is unique for each user). This, however, is somewhat darkened by SID being stored in the registry along with the master key.
- This algorithm, although it was created over 10 years ago, is firm against the so popular these days attacks using rainbow tables.

The suggested encryption routine, which, by the way, was being used already when Internet Explorer 4 was released, was a great breakthrough in data encryption. All modern browsers (Opera, Mozilla, Firefox, and Internet Explorer, up through version 6) use similar password encryption schemes. Take a note that the best Outlook's strongest competitors' creative imagination has stopped here, in the Middle Ages of encryption technology.

4.4 技术进步时代

技术进步时代--Outlook 2003正走在世界的前面。这个流行的电子邮件客户端的新版本使用了一种新的加密算法，它继续并在逻辑上发展了旧的加密算法。这种算法是基于一个重要的项目--它与用户

的密码绑定。这篇文章并不是要描述该算法的功能细节，因为这将需要相当多的篇幅。相反，我们只想说，要恢复用这种算法加密的密码，至少需要知道三件事（图3）：

1. 用户的主密钥
2. 用户的SID
3. 用户的密码

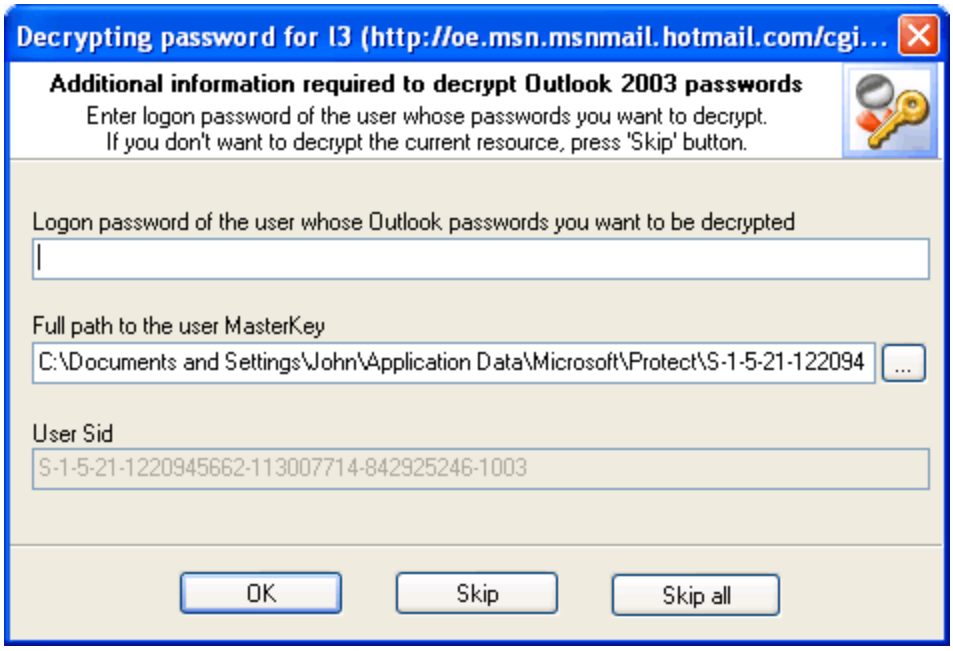

图3. Outlook 2003电子邮件帐户密码恢复

简而言之，新的DPAPI算法的优点是：

- 主密钥现在存储在本地计算机上的一个单独文件夹中。对该文件夹的访问受到部分限制。
- 主密钥的长度为512比特，这消除了在最近的将来挑选匹配的可能性。
- 使用新的加密算法，特别是SHA-HMAC，它在循环中使用可变的迭代次数（默认为4000）。
- 加密算法（用于主密钥和实际数据）是完全可定制的。人们可以设置操作系统所支持的任何属性。
- 数据保护可以通过操作系统的权限级别来实现。
- 该算法与用户的登录密码绑定。

对登录用户的密码进行加密是绝对透明的。当用户登录到操作系统时，只要求提供一次密码。操作系统负责其余的工作。即使一个潜在的黑客获得了对加密数据的物理访问，他也无法解密这些数据，除非他知道用户的密码。

5 结论

我们看到，随着这个流行的电子邮件客户端的每个新版本的出现，微软引入了一些新的东西，直到现在还不知道，从而证明他们确实关心最终用户的安全。新的Outlook 2003的加密机制是特别伟大的。

难道你不好奇，下一个版本，Outlook 2007的发布会发现什么？让我们冒昧地假设，现在存储在Windows的注册表中的PST密码，以及另外加密的方式，他们曾经在Outlook 9x（我们在谈论石器时代的时候涉及到这种方法）将被加密，因为他们现在在Outlook 2003中被加密了。至少，这将是符合逻辑的。对于电子邮件账户的密码，很难预测一些东西。我们最有可能看到新的基于Outlook 2003

的DPAPI加密算法的进一步发展，或者一个新的受保护的存储+用户密码的捆绑。无论如何，我们将期待这个流行的应用程序的新版本的发布，我们肯定会在接下来的文章中讲述程序中使用的加密机制。