LSA secrets in Windows

Passcape Software

内容

1 LSA secrets是什么？

LSA secrets是Windows中本地安全局（LSA）使用的重要数据的特殊保护存储。LSA被设计用来管理系统的本地安全策略、审计、验证、记录用户进入系统、存储私人数据。用户和系统的敏感数据被存储在秘密中。只有系统可以访问所有秘密数据。然而，如下所示，一些程序，特别是Windows Password Recovery,，允许覆盖这一限制。

2 LSA secrets中存储了什么？

最初，该secrets包含了缓存的域名记录。后来，Windows开发者扩大了存储的应用范围。此刻，他们可以存储PC用户的文本密码、服务账户密码（例如，那些必须由某个用户运行才能执行的任务）、Internet Explorer密码、RAS连接密码、SQL和CISCO密码、SYSTEM账户密码、EFS加密密钥等私人用户数据，以及更多。

例如，NL$KM秘密包含缓存的域密码加密密钥。L$RTMTIMEBOMB存储了距离Windows未激活副本到期的时间。L$HYDRAENCKEY存储远程桌面协议中使用的公共RSA2密钥。顺便说一下，即使没有设置自动登录，在某些版本的Windows 7中，秘密可能包含管理员账户密码的明文，从而危及整个目标系统。

3 LSA secrets存储在哪里？

LSA secrets以加密的形式存储在Windows注册表的HKEY_LOCAL_MACHINE/Security/Policy/Secrets键中。父键，HKEY_LOCAL_MACHINE/Security/Policy，包含访问和解密秘密所需的额外数据。下面是这个键的一些值的描述。

Key: **HKEY_LOCAL_MACHINE/Security/Policy/SecDesc**
Value name:
Data type: **REG_BINARY**
Description: security descriptor for accessing the registry tree with secrets.

Key: **HKEY_LOCAL_MACHINE/Security/Policy/PolState**
Value name:
Data type: **REG_BINARY**
Description: current state of the secrets subsystem.

Key: **HKEY_LOCAL_MACHINE/Security/Policy/PolRevesion**
Value name:
Data type: **REG_BINARY**
Description: contains the version of the subsystem.

Key: **HKEY_LOCAL_MACHINE/Security/Policy/PolPrDmS**
Value name:
Data type: **REG_BINARY**
Description: domain SID.

Key: **HKEY_LOCAL_MACHINE/Security/PolicyPolPrDmN**

Value name:
Data type: **REG_BINARY**
Description: domain name.

Key: **HKEY_LOCAL_MACHINE/Security/PolicyPolEKList**
Value name:
Data type: **REG_BINARY**
Description: contains the list of encryption keys for LSA secrets.

PolRevesion中的值1.1符合NT操作系统，1.5--Windows 2000，1.7--Windows XP和Win2K3，1.9--Windows Vista，1.10--Windows 7。在Windows Vista之前，只有一个加密密钥被存储在注册表中，在PolSecretEncryptionKey值中。从Windows Vista开始，PolEKList可以包含多个加密密钥。

4 LSA secrets细节

在物理层面上，秘密被存储在一个二进制注册文件SECURITY中，其中有钥匙的秘密名称。例如，Security/Policy/Secrets/$MACHINE.ACC。注册表中的每个秘密都由五个值表示：

1. CurrVal - 秘密的当前加密值。
2. CupdTime - 最后的更新时间，是一个8字节的FILETIME结构。
3. OldVal - 秘密的先前值。
4. OupdTime - 以前的更新时间。
5. SecDesc - 安全描述符，即哪些用户可以访问该秘密，哪些被禁止访问。

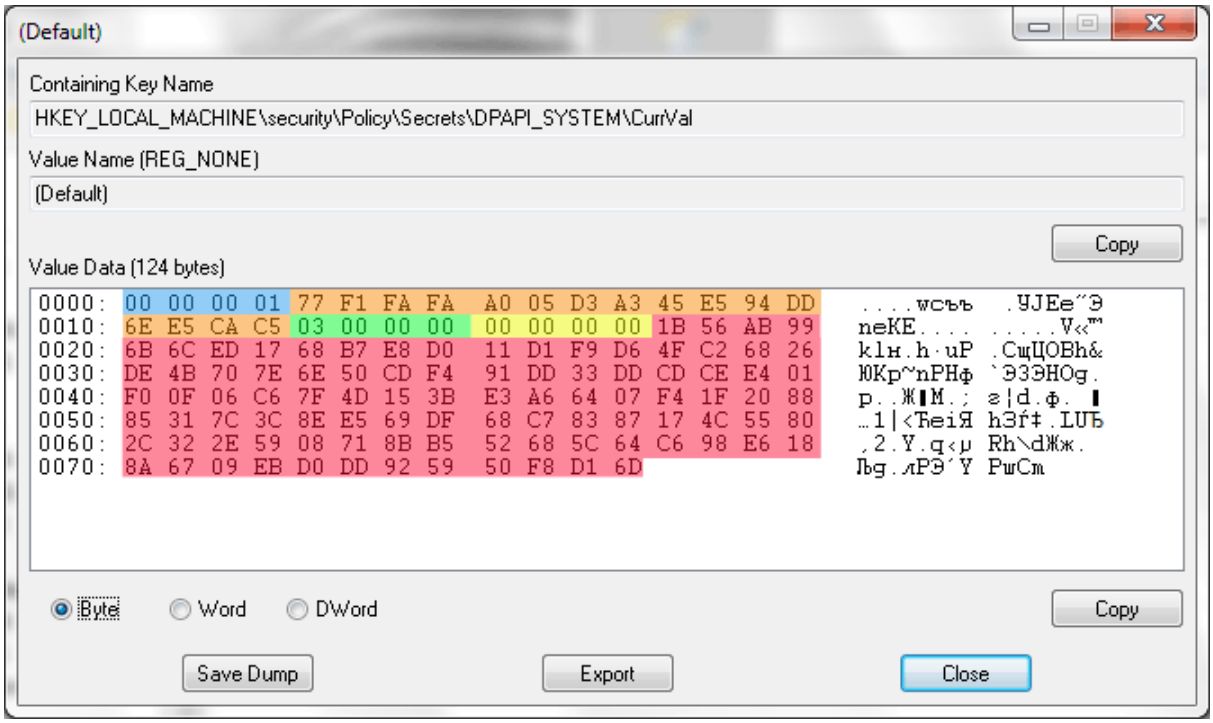
如果系统无法读取/解密其中一个秘密，它会在其中写入第六个值PolMod，这表明该秘密已损坏。例如，如果由于断电或注册表文件损坏，LSA数据库的事务未完成。

5 CurrVal和OldVal数据结构

从1.9版本开始，秘密的结构发生了巨大的变化；因此，我们不打算介绍旧的格式。现在，你可以将每个秘密与加密密钥列表（PolEKList）上的任何值绑定，而不是单一的加密密钥。

还有一个选项，可以选择一种加密算法！因此，数据结构中的前4个字节是数据的版本；然后是一个16字节的加密密钥标识符，用于在列表中找到必要的密钥。之后是一个DWORD，其标识符是秘密加密算法的列表。

例如，值3与SHA-256散列算法和AES-256块状加密算法的组合相匹配。算法标识符之后是一个4字节的值，其中有解密时使用的不同标志。最后，是加密的数据。请看该图。



6 Windows 2000中的LSA机密加密，XP，2003

在Windows Vista之前，解密这些秘密看起来相当琐碎。首先，需要解密秘密加密密钥。它看起来是这样的：

```
BOOL CSecrets::DecryptPrimaryKey()
{
    BYTE rc4key[0x10];

    MD5Init();
    MD5Update(m_pSyskey,0x10);
    for ( int i=0; i<1000; i++)
        MD5Update(((LPBYTE)m_pCypherKey)+0x3C,0x10);
    MD5Final(rc4key);

    RC4SetKey(rc4key,0x10);
    RC4Decrypt(((LPBYTE)m_pCypherKey)+0xC,0x30);

    return ( memcmp(((LPBYTE)m_pCypherKey)+0xC,CYPHERKEY_AUTHENTIFICATOR,0x10)==0 );
}
```

其中m_pSyskey - 16字节的SYSKEY值：
m_pCypherKey - 注册表键HKEY_LOCAL_MACHINE/Security/Policy/PolSecretEncryptionKey的值。

一旦获得了secrets的加密密钥，就可以进行secrets的解密了。secrets是用DES算法加密的。

7 Windows Vista和更高版本操作系统中的Lsa机密加密

在Windows Vista（和更高版本的操作系统）中，如前所述，加密算法变得更加复杂。首先，仍然需要解密存储在HKEY_LOCAL_MACHINE/Security/Policy/PolEKList中的加密密钥列表（是，现在允许多个密钥）。然后继续实际的秘密。每个秘密现在存储密钥标识符、加密算法标识符和实际加密数据。解密密钥的工作算法如下：

- 读取密钥值并找到加密密钥的标识符。
- 在加密密钥列表（PolEKList）中，使用你之前获得的标识符找到必要的密钥。
- 使用算法标识符和找到的密钥对secrets进行解密。

因此，LSA数据库中的secrets不仅可以用不同的算法进行加密，还可以有不同的原始背景。例如，使用其他PC的SYSKEY。

8 阅读和编辑secrets

有一套处理秘密的API供软件开发者使用。因此，任何Windows应用程序都可以创建和读取自己的秘密，但只能在当前用户上下文的范围内。请参阅附录1，其中有读取秘密的源代码。

如果你需要查看或编辑LSA的秘密，例如，删除你账户的文本密码，你可以利用 Windows Password Recovery tool，它有一个处理LSAsecrets的便捷插件。顺便说一下，这个插件既能处理当前操作系统的secrets，也能处理外部注册表文件。

9 附录

读取LSAsecrets的程序的源代码。请注意，并非所有的秘密都可以在用户环境下读取。此外，还需要管理员的权限。该程序的可执行文件可以在以下网址下载 following link.

```
// LsaSecretReader.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include <windows.h>
#include <stdio.h>
#include <ntsecapi.h>


#pragma comment (lib, "Advapi32")

PLSA_UNICODE_STRING InitLsaString(LPWSTR wszString, PLSA_UNICODE_STRING lsastr)
{
        if ( !lsastr )
                return NULL;
```

```
	if ( wszString )
	{
		lsastr->Buffer=wszString;
		lsastr->Length=(USHORT)lstrlenW(wszString)*sizeof(WCHAR);
		lsastr->MaximumLength=lsastr->Length+2;
	}
	else
	{
		lsastr->Buffer=L"";
		lsastr->Length=0;
		lsastr->MaximumLength=2;
	}

	return lsastr;
}


int _tmain(int argc, _TCHAR* argv[])
{
	NTSTATUS status;
	LSA_OBJECT_ATTRIBUTES att;
	LSA_HANDLE pol;
	LSA_UNICODE_STRING secret, *data=NULL;

	if ( argc!=2 )
	{
		_tprintf(TEXT("Syntax: %s secretnamen"),argv[0]);
		return 1;
	}

	memset(&att,0,sizeof(att));

	status=LsaOpenPolicy(NULL,&att,0,&pol);
	if ( status!=ERROR_SUCCESS )
	{
		_tprintf(TEXT("LsaOpenPolicy error: %lXn"),status);
		return 2;
	}

	InitLsaString(argv[1],&secret);
	status=LsaRetrievePrivateData(pol,&secret,&data);
	if ( status!=ERROR_SUCCESS )
	{
		_tprintf(TEXT("LsaRetrievePrivateData error: %lXn"),status);
		return 3;
	}
	LsaClose(pol);

	if ( data && data->Buffer && data->Length )
	{
		for ( USHORT i=0; i<data->Length; i+=16 )
```

```
                {
                        _tprintf(TEXT("%04X: "),i);
                        LPBYTE ptr=(LPBYTE)data->Buffer;
                        ptr+=i;
                        for ( int j=0; j<min(16,data->Length-i); j++ )
                                _tprintf(TEXT("%02X "),ptr[j]);
                        _tprintf(TEXT("n"));
                }
        }
        else
        {
                _tprintf(TEXT("No data"));
        }

        return 0;
}
```

输出示例

```
C:>LsaSecretReader.exe DPAPI_SYSTEM
0000: 01 00 00 00 73 4F 19 CF 6B B7 6C 8A BC 6D 35 EF
0010: 19 9C A6 3E 9A 80 A7 0C 9D D4 FD B1 20 C6 B1 A5
0020: 7A 87 5F 2B 51 3E 1D E0 45 9B 99 B2
```