LSA секреты Windows

Passcape Software

1 Что такое секреты LSA

LSA секреты - это специальное защищенное хранилище важной информации, которая используется системой Local Security Authority (LSA) в Windows. LSA предназначена для управления локальной политикой безопасности системы, аудита, авторизации, входа пользователей в систему, хранения приватных данных. Сенситивные данные пользователей и системы хранятся в секретах. Доступ ко всем секретным данным имеет только система. Однако, как показано ниже, некоторые программы, в частности Windows Password Recovery, позволяют обойти это ограничение.

2 Что находится в LSA секретах

Изначально в секретах хранились кэшированные записи домена. Затем, разработчики Windows расширили область применения этого хранилища. В настоящий момент в них могут храниться текстовые пароли пользователей ПК, пароли учетных записей служб (например, требующие запуска в контексте локального пользователя для выполнения определенных задач), пароли Internet Explorer, пароли сетевых подключений RAS, пароли SQL, CISCO, пароли учетной записи SYSTEM, приватные данные пользователя, например, ключи шифрования и многое другое.

Например, секрет с именем **NL$KM** содержит ключ шифрования кэшированных паролей домена. В **L$RTMTIMEBOMB** хранится время до окончания работы неактивированной копии Windows. **L$HYDRAENCKEY** хранит публичный RSA2 ключ, используемый в Remote Desktop Protocol. Между прочем, даже несмотря на то, что автоматический вход в систему не установлен, в секретах на некоторых версиях Windows 7 может находиться текстовый пароль учетной записи администратора, компрометируя таким образом целевую систему.

3 Где хранятся LSA секреты

LSA секреты расположены в зашифрованном виде в реестре Windows, в ключе **HKLM/Security/Policy/Secrets**. Родительский ключ **HKLM/Security/Policy** содержит дополнительные дочерние ключи с данными, необходимыми для доступа и расшифровки секретов. Ниже даны описания некоторых значений этих ключей.

Ключ: **HKEY_LOCAL_MACHINE/Security/Policy/SecDesc**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: описатель безопасности для доступа к ветке реестра с секретами

Ключ: **HKEY_LOCAL_MACHINE/Security/Policy/PolState**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: текущее состояние подсистемы с секретами

Ключ: **HKEY_LOCAL_MACHINE/Security/Policy/PolRevesion**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: содержит версию этой подсистемы

Ключ: **HKEY_LOCAL_MACHINE/Security/Policy/PolPrDmS**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: SID домена

Ключ: **HKEY_LOCAL_MACHINE/Security/PolicyPolPrDmN**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: имя домена

Ключ: **HKEY_LOCAL_MACHINE/Security/PolicyPolEKList**
Имя параметра:
Тип данных: **REG_BINARY**
Описание: содержит список ключей шифрования для секретов LSA

Из перечисленных ключей нас особенно интересует **PolRevision**, от которого зависит как расшифровывать данные. Значение 1.1 соответствует операционной системе NT, 1.5 - Windows 2000, 1.7 - Windows XP и Win2K3, 1.9 - Windows Vista, 1.10 - Windows 7. Основной ключ шифрования данных до Windows Vista хранился в ключе **PolSecretEncryptionKey**. Начиная с Windows Vista **PolEKList** может содержать несколько ключей. Это означает, что теоретически в хранилище можно записывать и читать секреты другого компьютера.

4 LSA secrets в деталях

На физическом уровне, секреты хранятся в двоичном файле реестра SECURITY и именем секрета в качестве ключа. Например, **Security/Policy/Secrets/$MACHINE.ACC**. Каждый секрет в реестре представлен пятью значениями:

1. **CurrVal** - текущее зашифрованное значение секрета
2. **CupdTime** - дата последнего обновления в виде 8 байтовой структуры FILETIME
3. **OldVal** - предыдущее значение секрета
4. **OupdTime** - дата предыдущего обновления
5. **SecDesc** - описатель безопасности секрета, т.е. какие пользователи имеют доступ к секрету, а каким он запрещен.

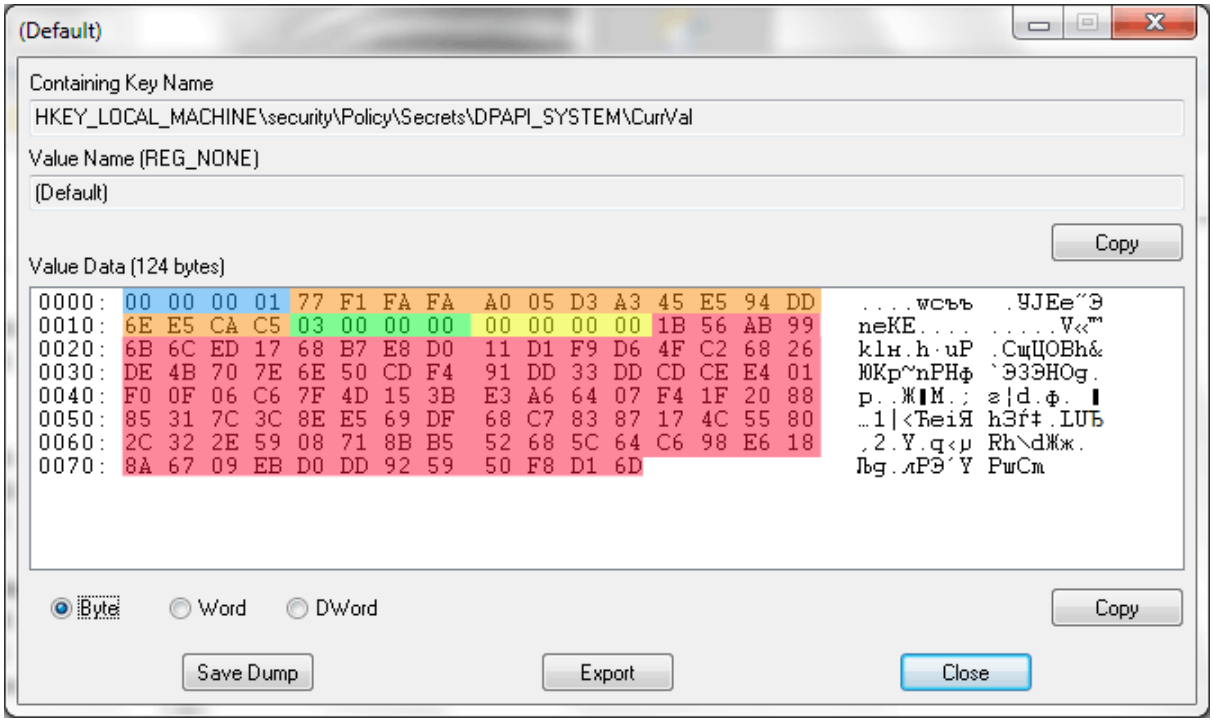
Если система не может прочитать/расшифровать один из секретов, то в нем прописывается шестое значение **PolMod**, означающая, что этот секрет испорчен. Например, транзакция в базу LSA не была завершена по причине сбоя электропитания или из-за повреждения файла реестра.

5 Структура данных CurrVal и OldVal

Начиная с версии 1.9, структура секретов поменялась кардинально, поэтому старый формат мы рассматривать не будем. Вместо одного ключа шифрования, теперь можно привязывать каждый секрет к любому значению в списке ключей шифрования (**PolEKList**). Появилась также возможность выбора алгоритмов шифрования!

Итак, первые 4 байта в структуре данных это версия данных, затем идет 16 байтный идентификатор ключа шифрования для поиска необходимого ключа в списке. За ним следует DWORD с идентификатором набора алгоритмов шифрования, с помощью которых зашифрован этот секрет.

Например, значение 3 соответствует связке из алгоритма хэширования **SHA-256** и алгоритма блочного шифрования **AES-256**. После идентификатора алгоритма идет 4 байтовое значение с различными флагами, используемыми при расшифровке. Ну и, наконец, сами зашифрованные данные. Смотрите рисунок.



6 Шифрование секретов LSA в Windows 2000/XP/2003

Расшифровка секретов вплоть до Windows Vista выглядела довольно тривиально. Сначала было необходимо расшифровать первичный ключ шифрования секретов. Выглядело это так:

```
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 );
}
```

Где **m_pSyskey** - 16 байтное значение **SYSKEY**
m_pCypherKey - значение из ключа реестра
HKLM/Security/Policy/PolSecretEncryptionKey

После того, как ключ шифрования секретов получен, можно приступать к расшифровке самих секретов, которые зашифрованы при помощи алгоритма **DES**. В качестве ключа шифрования - первичный ключ, полученый выше.

7 Шифрование секретов в Windows Vista и более поздних ОС

Начиная с Windows Vista, алгоритм шифрования, как уже было сказано, значительно усложнился. Вначале все также необходимо расшифровать список ключей шифрования (да, теперь их может быть несколько), хранящийся в **HKLM/Security/Policy/PolEKList**. Затем сами секреты. В каждом секрете теперь хранится идентификатор ключа, идентификатор алгоритма шифрования и сами зашифрованные данные. Алгоритм работы для расшифровки ключа выглядит примерно так:

- Чтение значение ключа и находим идентификатор ключа шифрования
- Поиск в списке ключей шифрования (PolEKList) нужного ключа, с помощью полученного ранее идентификатора
- Расшифровка секрета используя идентификатор алгоритма и найденный ключ

Таким образом, секреты в базе LSA теперь могут быть не только зашифрованы разными алгоритмами, но и разным исходным контекстом. Например, используя SYSKEY другой машины.

8 Чтение и редактирование секретов

Для разработчиков ПО существует набор API для работы с секретами. Так что любая программа в Windows может создать и читать свои собственные секреты, но только в рамках текущего контекста пользователя. Смотрите приложение 1 с исходным кодом чтения секретов.

Если вам требуется просмотр или редактирование LSA секретов, например, для удаления текстового пароля вашей учетной записи, то можно воспользоваться программой Windows Password Recovery, в которой есть удобный плагин для работы с LSA секретами. Этот плагин работает не только с секретами текущей операционной системы, но и с внешними файлами реестра.

9 Приложение

Исходный код программы для чтения секретов LSA. Имейте в виду, что не все секреты могут быть прочитаны в рамках пользователя ОС. Кроме того, наличие привилегий Администратора обязательно. Готовую программу можно скачать по этой ссылке.

```
// 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;
}
```

Пример вывода

```
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
```