(9) X

x.com/0gtweet/article/1848821825813315901

Reading BitLocker numerical passwords via API

22K

The technique is not crossing any security boundaries. Manage-bde.exe is a well-known tool and if you can use it at any moment. There are some WMI interfaces as well.

To read numerical keys you need to be an admin for fixed drives and regular user for removable ones.

In general, accessing BitLocker API happens through functions exported from fveapi.dll. For reading numerical password (a.k.a. Recovery Password or RP) the sequence of tasks is:

1. Call FveSetAllowKeyExport(TRUE). It allows you to access keys in the cleartext form. Lack of this call causes the subsequent calls to return 0x80310038 (FVE_E_FIPS_PREVENTS_EXTERNAL_KEY_EXPORT – "The Group Policy setting requiring FIPS compliance prevents the recovery password from being saved to Active Directory. When operating in FIPS-compliant mode, BitLocker recovery options can be either a recovery key stored on a USB drive or recovery through a data recovery agent. Check your Group Policy settings configuration.") [Line #162]

2. Call FveOpenVolumeW(), providing the following parameters [Line #169]:

a. PWSTR: Volume device such as "\\.\c:"

b. BOOL: Write access (?)

c. PHANDLE: handle to the volume, used to make next calls. Handle can be closed with FveCloseHandle() or FveCloseVolume() call.

The FveOpenVolumeW() function returns HRESULT value indicating the success (S_OK) or error reason such as 0x80070002 for wrong device name etc.

As the next calls can return different numbers of results, the typical Windows API approach is used: the first function call returns the number of elements (or bytes) to be returned, and the subsequent calls should be ready for it.

3. FveGetAuthMethodGuids() is called to obtain number of Key Protectors, using following parameters [Line #178]:

a. HANDLE: handle to the volume obtained earlier

b. LPGUID: pointer to the memory to be filled with GUIDs. NULL for querying

c. UINT: maximum number of GUIDs to return. 0 for querying

d. Pointer to UINT to get total number of GUID

After allocating the appropriate memory, the FveGetAuthMethodGuids() function is called again to fill the memory with real GUIDs [Line #191].

Now we can iterate through GUIDs. To obtain the detailed information about each key protector we need to:

4. Prepare the Auth Info structure members

a. ULONG: Size MUST be 56 – sizeof(Auth Info structure)

b. ULONG: Version MUST be 1

c. ULONG: Flags equals 1 to make query with GUID

d. GUID: GUID of the Key Protector to query about

5. Call FveGetAuthMethodInformation() specifying [Line #211]:

a. HANDLE: the handle to volume obtained earlier

b. PVOID: pointer to the Auth Info structure prepared above

c. SIZE_T: size of the buffer for the information returned

d. SIZE_T*: pointer to SIZE_T to obtain number of bytes required

The function call should return ERROR_INSUFFICIENT_BUFFER.

As now we have the number of bytes really required for Auth Info, we can allocate memory and call again the same function again [Line #230] using the same parameters and appropriate amount of memory. The resulting buffer contains the same information we have set for querying plus:

a. Number of "Elements"

b. Pointer to "Elements"

c. PWSTR: Description

d. FILETIME: Key Protector creation time

Elements are variable size structures containing the data about the key protector. Each element consists of:

a. Size

b. Version

c. Flags

d. ULONG: Type of the Key Protector. 1 for Recovery Passwords

e. BYTE[]: real data

If the Type equals 1 (Recovery Password Key Protector), we can re-query for the same GUID, using 0x00080002 as Flags and repeating well-known sequence: query to get the size [Line #250], allocate memory, query to get the data [Line #269]. Element’s data contains 8 UINT values. We need to multiply each value by 11 and we have got the recovery key.

PoC is available at
https://github.com/gtworek/PSBits/blob/master/Misc2/FveKeyProtectorsDump.c

-