RE Windows

filip.kafka@eset.sk

Základy reverzného inžinierstva

Literatúra

Practical Malware Analysis:

- Chapter 5: IDA Pro
- Chapter 8: Debugging
- Chapter 9: OllyDbg
- Chapter 11: Malware Behavior

The IDA Pro Book, Chris Eagle (1st edition - 2008, 2nd edition - 2011)

Prológ

- Častý začiatok funkcie
- Vytvára nový stack frame, alokácia lokálnych premenných, odloženie registrov na stack

Prológ

```
push    ebp                 ; Save ebp
mov     ebp, esp            ; Set stack frame pointer
sub     esp, localbytes     ; Allocate space for locals
push    <registers>         ; Save registers
```

Epilóg

- Častý koniec funkcie
- Obnovenie registrov zo stacku, obnovenie predošlého stack frame, návrat z funkcie (prechod EIP na adresu zo stacku)

Epilóg

```
pop     <registers>     ; Restore registers
mov     esp, ebp        ; Restore stack pointer
pop     ebp             ; Restore ebp
ret                     ; Return from function
```

MSDN - Windows API funkcie

- Popis
- Argumenty
- In, Out, opt
- Return value
- Remarks
- Súvisiace API

ENJOY SAFER TECHNOLOGY™

ReadFile function

Reads data from the specified file or input/output (I/O) device. Reads occur at the position specified by the file pointer if supported by the device.

This function is designed for both synchronous and asynchronous operations. For a similar function designed solely for asynchronous operation, see **ReadFileEx**.

Syntax

C++

```
BOOL WINAPI ReadFile(
  _In_        HANDLE       hFile,
  _Out_       LPVOID       lpBuffer,
  _In_        DWORD        nNumberOfBytesToRead,
  _Out_opt_   LPDWORD      lpNumberOfBytesRead,
  _Inout_opt_ LPOVERLAPPED lpOverlapped
);
```

Parameters

hFile [in]
A handle to the device (for example, a file, file stream, physical disk, volume, console buffer, tape drive, socket, communications resource, mailslot, or pipe).

The *hFile* parameter must have been created with read access. For more information, see Generic Access Rights and File Security and Access Rights.

For asynchronous read operations, *hFile* can be any handle that is opened with the **FILE_FLAG_OVERLAPPED** flag by the **CreateFile** function, or a socket handle returned by the **socket** or **accept** function.

lpBuffer [out]
A pointer to the buffer that receives the data read from a file or device.

This buffer must remain valid for the duration of the read operation. The caller must not use this buffer until the read operation is completed.

nNumberOfBytesToRead [in]
The maximum number of bytes to be read.

lpNumberOfBytesRead [out, optional]
A pointer to the variable that receives the number of bytes read when using a synchronous *hFile* parameter. **ReadFile** sets this value to zero before doing ar
work or error checking. Use **NULL** for this parameter if this is an asynchronous operation to avoid potentially erroneous results.

This parameter can be **NULL** only when the *lpOverlapped* parameter is not **NULL**.

For more information, see the Remarks section.

lpOverlapped [in, out, optional]
A pointer to an **OVERLAPPED** structure is required if the *hFile* parameter was opened with **FILE_FLAG_OVERLAPPED**, otherwise it can be **NULL**.

Často vyskytované API funkcie

Files

- CreateFile
- WriteFile
- ReadFile
- GetFileSize
- GetFileType
- SetFileAttributes
- FindFirstFile
- FindNextFile

Registry

- RegCreateKeyEx
- RegSetValue
- RegOpenKeyEx
- RegQueryValueEx
- RegCloseKey

Process

- CreateToolhelp32Snapshot
- Process32First
- Process32Next
- OpenProcess
- TerminateProcess
- GetCurrentProcess
- CreateProcess

Misc:

- GetModuleHandle
- GetModuleFileName
- ExitProcess
- LoadLibrary
- GetProcAddress
- GetLastError
- GetCommandLine
- CloseHandle
- GetComputerName
- GetUserName
- CreateMutex
- ExpandEnvironmentStrings
- GetDriveType
- GetVersionEx
- GetSystemTime
- GetTempPath
- GetTempFileName
- URLDownloadToFile

Win GUI:

- DialogBoxParam
- GetDlgItemText
- MessageBox
- FindWindow
- EndDialog

Network:

- WSAStartup
- Socket
- Inet_addr
- Htons
- Bind
- Listen
- Accept
- Connect
- Recv
- Send

Strings:

- lstrcat
- lstrlen
- lstrcmp
- lstrcpy

Anti-debug tricks:

- OutputDebugString
- GetTickCount
- IsDebuggerPresent

Štruktúry ako argumenty funkcii

FindFirstFile function

Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if v

To specify additional attributes to use in a search, use the **FindFirstFileEx** function.

To perform this operation as a transacted operation, use the **FindFirstFileTransacted** function.

Syntax

C++

```
HANDLE WINAPI FindFirstFile(
  _In_  LPCTSTR           lpFileName,
  _Out_ LPWIN32_FIND_DATA lpFindFileData
);
```

Parameters

lpFileName [in]

The directory or path, and the file name, which can include wildcard characters, for example, an aster

This parameter should not be **NULL**, an invalid string (for example, an empty string or a string that i backslash (\).

If the string ends with a wildcard, period (.), or directory name, the user must have access permissior

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this lim the function and prepend "\\?\" to the path. For more information, see Naming a File.

lpFindFileData [out]

A pointer to the **WIN32_FIND_DATA** structure that receives information about a found file or direct

WIN32_FIND_DATA structure

Contains information about the file that is found by the **FindFirstFile**, **FindFirstFileEx**, or **FindNextFile** function.

Syntax

C++

```
typedef struct _WIN32_FIND_DATA {
  DWORD    dwFileAttributes;
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD    nFileSizeHigh;
  DWORD    nFileSizeLow;
  DWORD    dwReserved0;
  DWORD    dwReserved1;
  TCHAR    cFileName[MAX_PATH];
  TCHAR    cAlternateFileName[14];
} WIN32_FIND_DATA, *PWIN32_FIND_DATA, *LPWIN32_FIND_DATA;
```

Windows API != Systémové volanie

C funkcie (ktoré musia interagovať s OS) -> …… Windows API -> …. Systémové volanie

- Musia interagovať s OS: fopen, scanf, puts - hlavne z knižnice stdio.h
- Nemusia interagovať s OS: dátové operácie - hlavne z knižnice string.h

Knižnice – .dll, .lib a .h

- .h: Header file, zdrojový súbor obsahujúci deklarácie (opak .cpp, .cxx... obsahujúce implementáciu).

- .lib: Static library môže obsahovať kód alebo odkazovať sa do .dll. V oboch prípadoch je to skompilovaný kód, ktorý sa linkuje k programu. Statická knižnica je pripojená k .exe súboru pri linkovaní.

- .dll: Dynamic-link library. Program z nej načítava funkcie pri spúštaní => je od nej závislý.

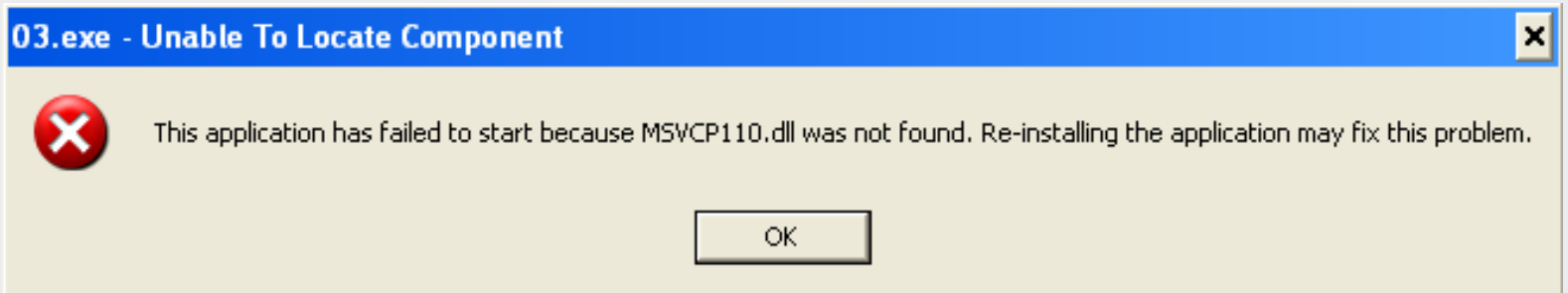
Run-Time loading (run-time dynamic linking)

- LoadLibrary, GetProcAddress
- Načítanie ďalšieho modulu do adresného priestoru procesu je možné aj počas behu programu
- Ak modul ešte nebol načítaný v procese, spustí jej DllMain s argumentom DLL_PROCESS_ATTACH
- Neodporúča sa volať LoadLibrary v DllMain

```
typedef HANDLE (WINAPI *FindFirstFileFcn)(LPCWSTR, LPWIN32_FIND_DATAW);

HMODULE hMod = LoadLibrary(TEXT("kernel32.dll"));
if (!hMod)
    return false;

FindFirstFileFcn pFindFirstFile = (FindFirstFileFcn) GetProcAddress(hMod, "FindFirstFileW");
if (pFindFirstFile == NULL)
    return false;

WIN32_FIND_DATAW findFileData = { 0 };
HANDLE hFind = pFindFirstFile(L"C:\\Windows\\*", &findFileData);
```

C Run-Time (CRT)

- Sada štandardných funkcií pre C/C++, napr.:
- Memory Allocation: malloc, realloc, new, delete, delete[], ... - HeapAlloc
- Data Conversion: atoi, toupper, ...
- I/O: fopen, fprintf, fflush, kbhit, ...
- String Manipulation: strtok, strncmp, strchr, strcpy, ...
- Error Handling, Sorting, Debugging Routines, ...
- Jednoduchý program vieme napísať aj bez závislosti na CRT
- CRT je možné linkovať staticky, alebo dynamicky (vytvorí závislosti na externých knižniciach)
- Tie môžu spôsobovať nečakané chybové hlásenia



Inline linking (C / C++)

Vkladanie kódu funkcie priamo do miesta, kde má byť funkcia volaná.

Výhody:

- Rýchlosť (Netreba prológ, epilóg, call, ret. Dá sa zoptimalizovať na mieru).

Nevýhody:

- Veľkosť (ak chcem volať funkciu viackrát, celý kód funkcie musí byť v programe viackrát).