ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Vulnerability Analysis and Tracking of Covert Malicious Software Launching

Surabhi Dubey

Master of Technology, Department of Computer Science &Engineering, Delhi Institute of Technology, Management &Research (Faridabad) affiliated to Maharshi Dayanand University, Rohtak – Haryana, India

ABSTRACT: here are a number of covert launching techniques but the most popular one is process injection. Process injection is a technique which injects malicious code into another running process and that process execute that malicious code. This technique conceals malicious behaviour of their code and may bypass firewalls and other process-specific security mechanisms. Certain Windows API calls are commonly used for process injection. For example, the VirtualAllocEx function can be used to allocate space in a remote process's memory, and WriteProcessMemory can be used to write data to that allocated space.

KEYWORDS: Dll Injection, Remothethread, Windows API, Process, Covert, Process replacement, Malicious, Hook Injection, IAT, Inline

I. INTRODUCTION

Covert malware launching techniques are the methods used by the malware authors to avoid detection of the Malware. The malwareauthor use different methods including DLL injection, Process,replacement,Hooking, APC injection, Detours, Alternate Data Streams (ADS) etc. to execute malicious code. These techniquesconceal malicious behaviour of their code and may bypass firewalland other process specific security mechanism [2]. Windows APIcalls are commonly used for process injection. For example, the VirtualAllocExfunction can be used to allocate space in anexternal process's memory, and WriteProcessMemorycan be used to write data to that allocated space there are two types of Processinjection: DLL injection and direct injection [4].

II. THEORY AND ANALYSIS

Among the various covert launching techniques most popular one is process injection [1].There are various methods to inject code into the address spaceof the remote process.

A. DLL Injection: DLL injection—a form of process injection where a remote process is forced to load a malicious DLL—is the most commonly usedcovert loading technique. DLL injection works by injecting codeinto a remote process that calls LoadLibrary, thereby forcing a DLL to be loaded in the context of that process. Once the compromisedprocess loads the malicious DLL, the OS automatically calls theDLL's DllMainfunction, which is defined by the author of the DLL [3]. This function contains the malicious code and has as muchaccess to the system as the process in which it is running.

Methods for Injecting DLL: The Window API offers a number of functions that allow attaching and manipulating into other program at run time. There are variousmethods for injecting DLL into remote process [6].

a) Using the system registry.
b) UsingCreateRemoteThread function
c) Using a CodeCave

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

In order to inject the malicious DLL into a host program, the launcher malware must first obtain a handle to the victim process.The most common way is to use the Windows API calls CreateTool help 32 Snapshot, Process 32 First, and Process 32 Next to search the process list for the injectiontarget. Once the target is found, the launcher retrieves the ProcessIdentifier (PID) of the target process and then uses it to obtainthe handle via a call to OpenProcessUsing CreateRemoteThreadfunction is commonly used for DLL injection to allow thelauncher malware to create and execute a new thread in a remoteprocess [5]. When CreateRemoteThreadis used, it is passed threeimportant parameters: the process handle (hProcess) obtained with OpenProcess, along with the starting point of the injected thread(lpStartAddress) and an argument for that thread (lpParameter).For example, the starting point might be set to LoadLibraryandthe malicious DLL name passed as the argument. This will triggerLoadLibraryto be run in the victim process with a parameter ofthe malicious DLL, thereby causing that DLL to be loaded in the victim process (assuming that LoadLibraryis available in thevictim process's memory space and that the malicious library name string exists within that same space).Malware authors generally use VirtualAllocExto create space for the malicious library name string. The VirtualAllocExfunction allocates spacein a remote process if a handle to that process is provided. The last setup function required before CreateRemoteThreadcan becalledis WriteProcessMemory. This function writes the maliciousDLL name string into the memory space that was allocated withVirtualAllocEx. [4].DLL injection using CreateRemoteThreadfunction.There are a number of steps involved given below:

1. Create the process by calling CreateProcess(……) Win32 API.

```
BOOL WINAPI CreateProcess(
  _In_opt_     LPCTSTR lpApplicationName,
  _Inout_opt_  LPTSTRlpCommandLine,
  _In_opt_  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  _In_opt_   LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_        BOOL bInheritHandles,
  _In_        DWORD dwCreationFlags,
  _In_opt_     LPVOID lpEnvironment,
  _In_opt_     LPCTSTR lpCurrentDirectory,
  _In_        LPSTARTUPINFO lpStartupInfo,
  _Out_   LPPROCESS_INFORMATION lpProcessInformation
);[8]
```

2. Attach the process by calling OpenProcess Win32 API.

```
 HANDLE WINAPI OpenProcess(
  _In_  DWORDdwDesiredAccess,
  _In_  BOOLbInheritHandle,
  _In_  DWORDdwProcessId);
```

3. Allocate memory within the process using VirtualAllocEx() win32 API

```
LPVOID WINAPI VirtualAllocEx(
            _In_ HANDLE hProcess,
            _In_opt_ LPVOIDlpAddress,
            _In_      SIZE_T dwSize,
            _In_      DWORD flAllocationType,
```

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

```
    _In_      DWORD flProtect
    );
```

4. Get the HANDLE for kernel32 library, to acquire LoadLibrary() routine address .The LoadLibrary() is a Kernel32.dll function used to load DLLs,executables and other support at run time.ittakes file name as its only parameter .It means that we just needto allocate some memory for the path of our DLL.

```
1. HMODULE WINAPI GetModuleHandle( _In_opt_ LPCTSTRlpModuleName);
2. FARPROC WINAPI GetProcAddress(_In_ HMODULEhModule,_In_ LPCSTRlpProcName);
```

5. Write DLL name into the Virtual address of the process, whichwe allocate using VirtualAllocEx.()

```
BOOL WriteProcessMemory(
  HANDLE hProcess,
  LPVOID lpBaseAddress,
  LPVOID lpBuffer,
  DWORD nSize,
  LPDWORD lpNumberOfBytesWritten
);
```

6. And finally, after all the prior steps are successfully complete,we are ready to call CreateRemoteThread() function to create theremote thread in the target process:[11]

```
HANDLE WINAPI CreateRemoteThread(
  _In_  HANDLE hProcess,
  _In_  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  _In_  SIZE_T dwStackSize,
  _In_  LPTHREAD_START_ROUTINE lpStartAddress,
  _In_  LPVOID lpParameter,
  _In_  DWORD dwCreationFlags,
  _Out_ LPDWORDlpThreadId
);
```

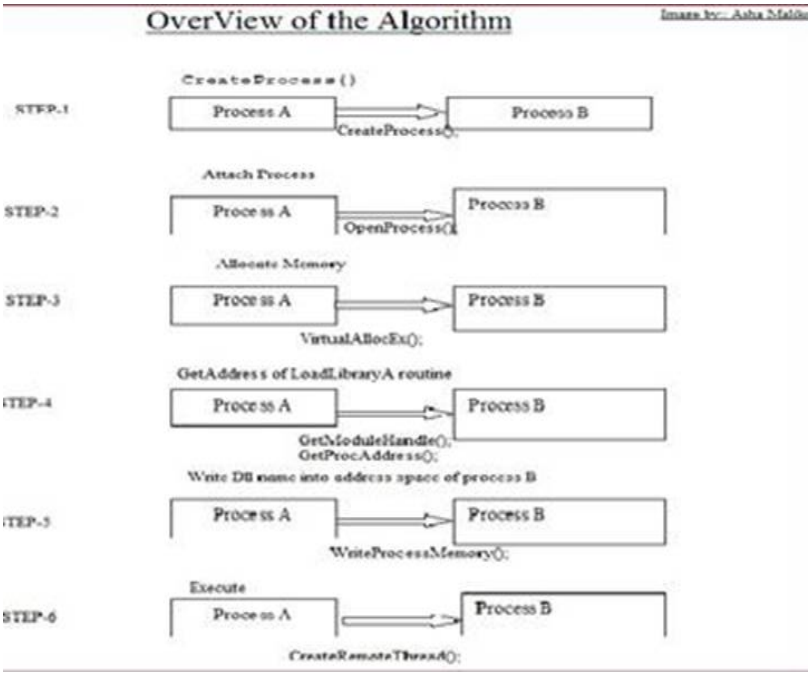
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017



B. Direct Injection: Like DLL injection, direct injection involves allocating and inserting codeinto the memory space of a remote process. Directinjection uses many ofthe same Windows API calls as DLL injection. The difference is that insteadof writing a separate DLL and forcing the remote process to load it; directinjectionmalware injects the malicious code directly into the remote process. Directinjection is more flexible than DLL injection. [8]Three functionsare commonly found in cases of direct injection:VirtualAllocEx,WriteProcessMemory, and CreateRemoteThread. There will typicallybe two calls to VirtualAllocExand WriteProcessMemory. The first will allocateand write the data used by the remote thread, and the second willallocate and write the remote thread code. Thecall to CreateRemoteThreadwill contains the location of the remote thread code (lpStartAddress) andthe data (lpParameter). Directinjectionrequires that authors either be skilled assembly languagecoders or that theywill inject only relatively simple shellcode.In order to analyse the remote thread's code, you may need todebugthe malware and dump all memory buffers that occur before calls toWriteProcessMemoryto be analysed in a disassembler.

C. Process Replacement Hide/Hollowing: Processreplacement involves running a process on the system andoverwriting its memory space with malicious executable.Processreplacement provides the malicious executable same privilege as the process it is replacing. So that the malware appear to beexecuting as legitimate process. But it leaves a fingerprint. The image in memory will differ from image in disk [5].Process Replacementhide the presence of a process. The idea ofProcess Hollowing is that a bootstrap application creates a process in a suspended state. The legitimate image is then unmapped andreplaced with the image that is to be hidden. Once the new imageis loaded in memory the EAX register of the suspended threadis set to the entry point. The process is then resumed and the entry point of the new image is executed. So the malware Firstcreates a process in a suspended state by calling CreateProcess then calls ZwUnmapViewSection to un-reserve the memory, allocatesmemory using VirtualAlloc, writes data to the process memoryusing WriteProcessMemory, retrieves the context of the threadusing GetThreadContext, modifies the context and then sets thecontext using SetThreadContext and then call ResumeThread tostart the process [9].

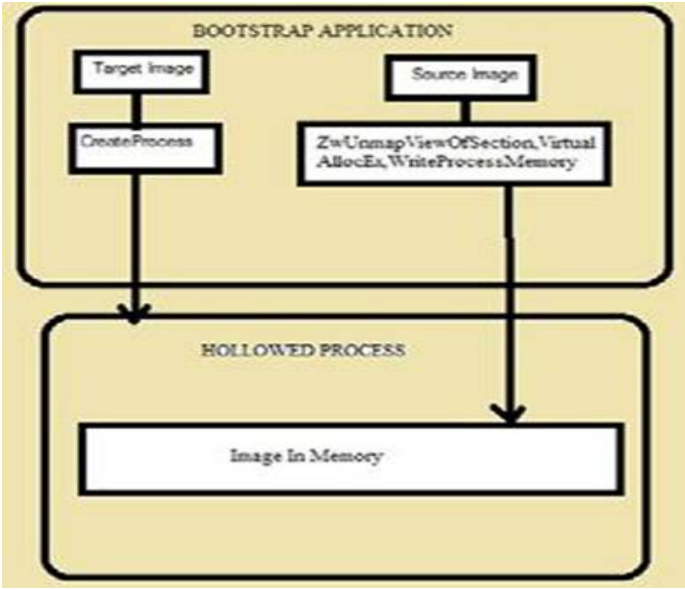
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017



Process Hollowing/Replacement involve following steps:

1. First, the malware starts a legitimate process using CreateProcess Win32API in suspended state by setting CREATE_SUSPENDED option in the fdwCreate flags parameter. The MSDN definitionof CreateProcess is:

   BOOL CreateProcess(LPCWSTR pszImageName, LPCWSTRpszCmdLine, LPSECURITY_ATTRIBUTES psaProcess,LPSECURITY_ATTRIBUTESpsaThread, BOOL fInheritHandles,DWORDfdwCreate, LPVOID pvEnvironment, LPWSTR pszCurDir, LPSTARTUPINFOW psiStartInfo, LPPROCESS_INFORMATION ProcInfo);

   The fdwCreate control the priority and creation of the process.CREATE_SUSPENDEDfdwCreate flag create primary threadof the new process is created in a suspended state, and does notrun until the ResumeThread function is called. The host program is now loaded but no code has been executedyet since it is started in suspended mode. The malware also hasa handle to the process it started through the ProcInfo structurepassed to CreateProcess.

2. While the host process is suspended, the malware first unmapsthe legitimate code from memory in the host process. The ZwUnmapViewOfSection WIN32 API function may be used tounmap the original code:
   The ZwUnmapViewOfSection routine unmaps a view of a section from the virtual address space of a subject process.
   NTSTATUS ZwUnmapViewOfSection(_In_ HANDLE ProcessHandle, _In_opt_ PVOID BaseAddress);
   Because the unmap function is a kernel API function, you willoften see the malware dynamically resolve it ‘s function addressat runtime as follows:
   HMODULE handle = GetModuleHandle(“ntdll.dll”);
   FARPROC loadLibraryAddress = GetProcAddress(handle,“ZwUnmapViewOfSection”);

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

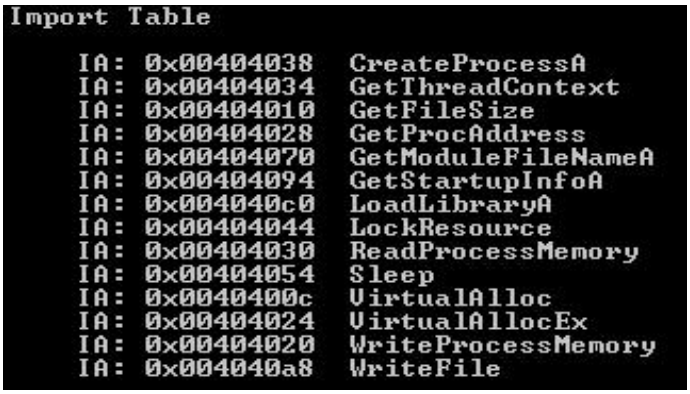
(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

3. The malware then allocates memory for the new code usingVirtualAllocEx Win32 API. It must ensure the code is marked as writeable and executable using the flProtect parameter. The malware can change this setting when it is done filling in the hollowed process memory.
   LPVOIDWINAPIVirtualAllocEx(_In_ HANDLEhProcess, _In_opt_ LPVOID lpAddress, _In_ SIZE_T dwSize, _In_ DWORDflAllocationType, _In_ DWORD flProtect);
4. The malware then writes its own new code into the hollow hostprocess using WriteProcessMemory, writing data to the memoryallocated in the host process with VirtualAllocEx.
   BOOL WriteProcessMemory(HANDLE hProcess, LPVOID lpBaseAddress, LPVOID lpBuffer, DWORD nSize, LPDWORD lpNumberOfBytesWritten);
5. If the malware author is careful, they will change the adjustcode and data sections look normal with Read/Execute or Readonly protections using VirtualProtectEx BOOL WINAPI VirtualProtectEx(__in HANDLE hProcess,__in LPVOID lpAddress, __in SIZE_T dwSize, __in DWORD flNewProtect, __out PDWORD lpflOldProtect);
6. The malware then adjusts the remote context to point to the new code section and may perform other cleanup tasks as necessary. The SetThreadContext function can be used to perform this step.
   BOOL WINAPI SetThreadContext(__in const CONTEXT*lpContext);
   Once everything is ready, the malware loader simply resumes thesuspended process using ResumeThread.
   DWORD WINAPI ResumeThread(__in HANDLE hThread);

D. Analysis of Process Replacement/Hollowing: In order to confirm the validity of the proposed method, it hasbeen first applied to the simulation experiment. Inthis simulationexperiment we take a malicious sample-Malicious.exe and analyse its process hollowing/replacement behaviour. Let's start ExeScanand Do static analysis of Malicious.exe. ExeScan shows following Imports [3]:



Now let's run the Malicious.exe by double-clicking its icon. Once run, Malicious.exe should be visible inside Process Explorer. In process explorer we further notice that it creates the sub process svchost.exe, and then exits, but leaves the svchost.exe process running as an orphaned process [7].(An orphaned process has no parent process listed in the process tree structure.) The fact that svchost.exe is orphaned is highly suspicious because svchost.exe is typically a child of services.exe.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Process	PID	CPU	Description	Company Name	User Name	Path	Version
csrss.exe	684		Client Server Runtime Process	Microsoft Corporation	NT AUTHORITY\...	C:\WINDOWS\sy...	5.1.2600.5512
DPCs	n/a		Deferred Procedure Calls				
explorer.exe	1688		Windows Explorer	Microsoft Corporation	ZINK-35DF2367B...	C:\WINDOWS\ex...	6.0.2900.5634
Interrupts	n/a	28.57	Hardware Interrupts				
lsass.exe	764		LSA Shell (Export Version)	Microsoft Corporation	NT AUTHORITY\...	C:\WINDOWS\sy...	5.1.2600.5512
procexp.exe	616		Sysinternals Process Explorer	Sysinternals - www.sysinter...	ZINK-35DF2367B...	C:\Documents an...	11.33.0.0
Procmon.exe	1112		Process Monitor	Sysinternals - www.sysinter...	ZINK-35DF2367B...	C:\Documents an...	3.5.0.0
rundll32.exe	1928		Run a DLL as an App	Microsoft Corporation	ZINK-35DF2367B...	C:\WINDOWS\sy...	5.1.2600.5512
services.exe	752	2.86	Services and Controller app	Microsoft Corporation	NT AUTHORITY\...	C:\WINDOWS\sy...	5.1.2600.5512
smss.exe	636		Windows NT Session Mana...	Microsoft Corporation	NT AUTHORITY\...	C:\WINDOWS\sy...	5.1.2600.5512
softinfo.exe	1988		Software Informer	Informer Technologies, Inc.	ZINK-35DF2367B...	C:\Program Files\...	1.2.918.0
SSScheduler.exe	2024		McAfee Security Scanner Sc...	McAfee, Inc.	ZINK-35DF2367B...	C:\Program Files\...	3.8.130.0
svchost.exe	1968		Generic Host Process for Wi...	Microsoft Corporation	ZINK-35DF2367B...	C:\WINDOWS\sy...	5.1.2600.5512
System	4				NT AUTHORITY\...		
System Idle Process	0	60.00			NT AUTHORITY\...		

Fig. 1: Showing svchost.exe process with PID 1968 as a orphaned process.

We investigate further by selecting Properties for svchost.exeprocess with PID 1968. From this same properties page, weselect Strings to show the strings in both the executable imageon disk and in memory. But the Executable Image on Memoryand on disk shows significant discrepancies. As shown in Figure2 the strings in memory contain practicalmalwareanalysis.log and [ENTER] but string in disk contain none of these two string.

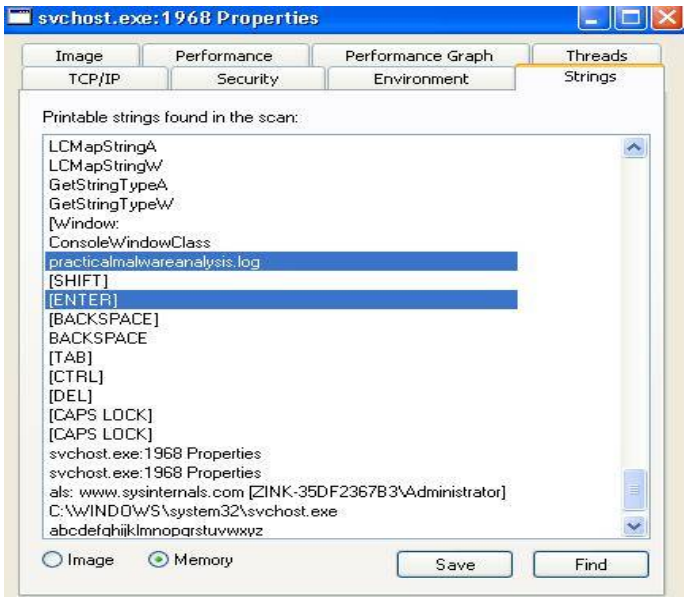


Fig. 2: Showing Executable Image in Memory

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

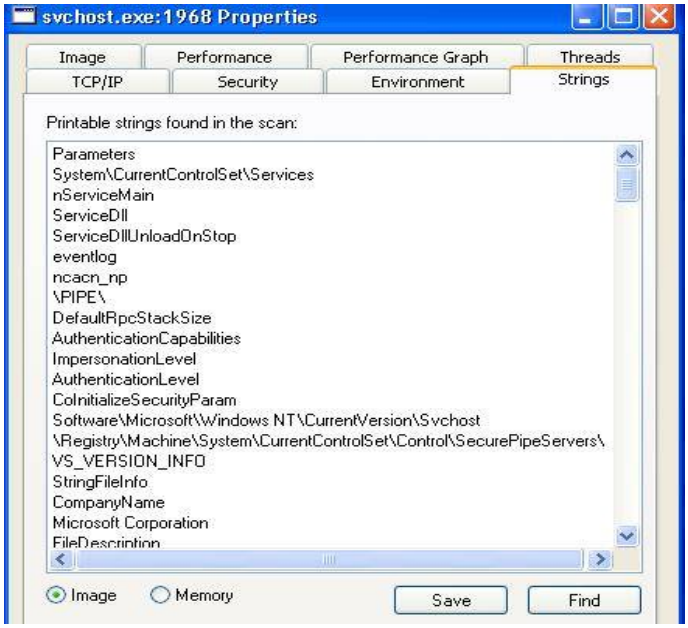
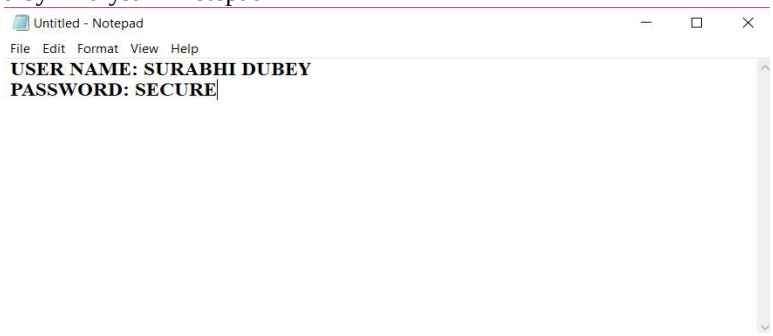


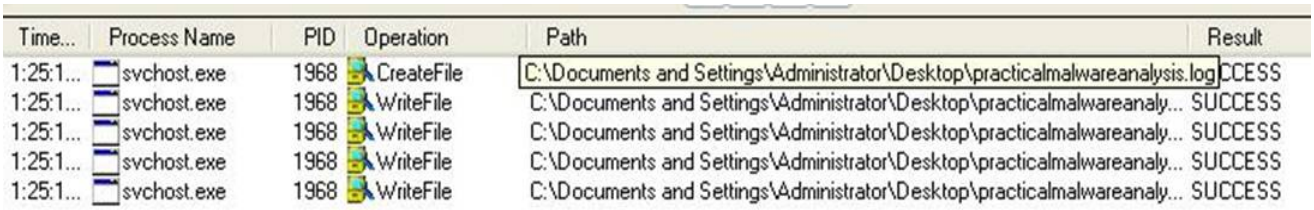
Fig. 3: Showing Executable Image in Disk

Showing Message Typed by Analyst in Notepad



The presence of the string practicalmalwareanalysis.log, coupledwith strings like [ENTER] and [CAPS LOCK], suggests that this program is a keylogger. To test our assumption, we open Notepad and type a short message to see if the malware will perform keylogging.

To analyse further , we use the PID 1968 (found in Process Explorer) for the orphaned svchost.exe to create a filter in procmon to show only events from that PID (1968).As you can see in Figure 4, the CreateFileand WriteFileevents for svchost.exe are writingto the file named practicalmalwareanalysis.log on desktop.



Time...	Process Name	PID	Operation	Path	Result
1:25:1...	svchost.exe	1968	CreateFile	C:\Documents and Settings\Administrator\Desktop\practicalmalwareanalysis.log	CCESS
1:25:1...	svchost.exe	1968	WriteFile	C:\Documents and Settings\Administrator\Desktop\practicalmalwareanaly...	SUCCESS
1:25:1...	svchost.exe	1968	WriteFile	C:\Documents and Settings\Administrator\Desktop\practicalmalwareanaly...	SUCCESS
1:25:1...	svchost.exe	1968	WriteFile	C:\Documents and Settings\Administrator\Desktop\practicalmalwareanaly...	SUCCESS
1:25:1...	svchost.exe	1968	WriteFile	C:\Documents and Settings\Administrator\Desktop\practicalmalwareanaly...	SUCCESS

Fig.4: Showing Filtered Activities of svchost.exe(replaced with malicious.exe) in Procomon

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

Now we open practicalmalwareanalysis.logfrom desktop and reveal the keystrokes we entered in Notepad. We conclude thatthis malware is a keylogger that uses process replacement on svchost.exe.malware performs process replacement on svchost.exe.

Comparing the disk image of svchost.exe with its memory image shows that they are not the same. The memory image has strings such as practicalmalwareanalysis.log and [ENTER], but the disk image has neither. The malware creates the log filepracticalmalwareanalysis.log. The program performs process replacement on svchost.exe to launch a keylogger.

E. Hook Injection: The term hooking represents a fundamental technique of gettingcontrol over a particular piece of code execution. It provides astraightforward mechanism that can easily alter the operating system's behaviour .When an inline hook is implemented it willoverwrite the first bytes codes of a windows api in order to redirect code flow. This kind of technique can be used in ring 3 or ring 0modes. Well-known and widespread malware take advantage ofthis kind of technique to hook key api windows components inorder to steal sensitive information from the user.Hooking is classified into two types:

1. User mode hooks
   - IAT (Import Address Table) Hooking
   - Inline Hooking
2. Kernel Mode hooks
   - IDT Hooking
   - SSDT Hooking etc. [1]

i. **IAT Hooking:**IAT hooking is a means to accesses code of a binarie's IAT andreplaces pointers which point to functions in linked dlls. By thismethod it is possible to hook a function, e.g. the MessageBoxA function and inject your own implementation for it.[1]

- Find out the File-offset of PE Header.
- Find Out VA of IMAGE_IMPORT_DESCRIPTOR Structure.
- Now we have functions with its real addresses and we canpatch our own code here:
- Now change the address of IAT function with your code address.

Hook injection describes a way to load malware that takes advantage ofWindowshooks, which are used to intercept messages destined for applications. Malware authors can use hook injectionto accomplish two things: To be sure that malicious code will run whenever a particular message isinterceptedTo be sure thata particular DLL will be loaded in a victim process'smemory space [4].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

ii. **Inline Hooking:**Implementing an inline hook: We inject our dll into the process we want to hijack. The first bytes of the target api are saved, these are the bytes that will beoverwritten. We place a jump, call or push-ret instruction. The jump redirects the code flow to our function. The hook can later callthe original api using the saved bytes of the hooked function. Theoriginal function will return control to our function and then datacan be easily tampered.[2]There are two types of Windows hooks:

- Local hooks are used to observe or manipulate messages destined for aninternal process.
- Remote hooks are used to observe or manipulate messagesdestined for a remote process (another process on the system).

iii. ***Using SetWindowsHookEx:*** The principal function call used to perform remote Windows hooking isSetWindowsHookEx, which has the following parameters:idHook specifies the type of hook procedure to install.lpfnPoints to the hook procedure.hModFor high-level hooks, identifies the handle to the DLL containingthe hook proceduredefined by lpfn. For low-level hooks, this identifies thelocal moduleinwhichthelpfnprocedureisdefined.dwThreadIdSpecifies the identifier of the thread with which the hookprocedure is tobe associated. If this parameter is zero, the hook procedure is associated with all existing threads running in the same desktopsthe calling thread. This must be set to zero for low-level hooks[9]. The hook procedure can contain code to process messagesas they comein from the system, or it can do nothing. Either way, the hook proceduremust call CallNextHookEx, which ensures that the next hook procedure in thecall chain gets the messageand that the system continues to run roperly.Assembly code forperforming hook injection inorder to load a DLL in a differentprocess's memory space.

```
push    esi
push    edi
push    offset LibFileName ; "hook.dll"
call    LoadLibraryA
mov     esi, eax
push    offset ProcName ; "MalwareProc"
push    esi             ; hModule
call    GetProcAddress
mov     edi, eax
call    GetNotepadThreadId
push    eax             ; dwThreadId
push    esi             ; hmod
push    edi             ; lpfn
push    WH_CBT   ; idHook
call    SetWindowsHookExA
```

The malicious DLL (hook.dll) is loaded by the malware, andthe malicious hook procedure address is obtained. The hook procedure,MalwareProc, calls only CallNextHookEx. SetWindowsHookExis then called for a threadin notepad.exe (assuming that notepad.exe is running). GetNotepadThreadIdisalocally defined function that obtains a dwThreadIdfor notepad.exe. Finally, aWH_CBT message is sent to the injected notepad.exe in order to force hook.dll tobe loaded by notepad.exe. Thisallows hook.dll to run in the notepad.exe processpace [1]. Oncehook.dll is injected, it can execute the full malicious code storedinDllMain, while disguised as the notepad.exe process. Since MalwareProcedure calls onlyCallNextHookEx, it should not interferewith incoming messages, but malware often immediately calls LoadLibraryandUnhookWindowsHookExinDllMaintoensurethat incoming messages are not impacted [7].

F. APC Injection: Windows supports Asynchronous Procedure Calls (APCs). An APC is a kernel-defined control object that represents a procedurethat is called asynchronously. APCs have the following features:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

- An APC always runs in a specific thread context.
- An APC runs at OS predetermined times.
- They can cause preemption of the currently running thread.
- APC routines can be preempted.

The Win32 API provides the QueueUserAPC function, which allows an application to queue an APC object to a thread. The queuing of an APC is a request for the thread to call the APC function. When a user-mode APC is queued, the thread is not directed to call the APC function unless it is in an alertable state. Unfortunately, a thread enters an alertable state only by using one of the following Win32 API functions:

SleepEx, SignalObjectAndWait, WaitForSingleObjectEx, WaitForMultipleObjectEx, orMsgWaitForMultipleObjectsEx.

APCs can direct a thread to execute some other code prior to executingits regular execution path. Every thread has a queue of APCs attached to it, and these are processed when the thread is in an alertable state, such as when they call functions like WaitForSingleObjectEx, WaitForMultipleObjectsEx,and Sleep [7]. These functions essentially give the thread a chance toprocess thewaiting APCs.If an application queues an APC whilethe thread is alertable but beforethe thread begins running, thethread begins by calling the APC function. A thread calls the APCfunctions one by one for allAPCs in its APC queue. When theAPC queue is complete, the thread continues running along itsregularexecution path. Malware authors use APCs to preempt threadsin analertable state in order to get immediate execution for theircode.APCs come in two forms:

- AnAPC generated for the system or a driver is called a kernel-mode APC.
- An APC generated for an application is called a user-modeAPC [8].

i. ***Procedure of DLL Injection using QueueUserAPC:***Get Handle of the remote process. By using API calls such asCreateToolhelp32Snapshot, Process32First, and Process32Next
   - Get some memory in the remote process via VirtualAllocEx
   - Write the name of the DLL which we want to inject to thismemory via WriteProcessMemory
   - Create an asynchronous procedure call on LoadLibraryA in the remote process

ii. ***APC Injection from User Space:***From user space, another thread can queue a function to be invoked in aremote thread, using the API function QueueUserAPC.Themsdn defines QueueUserAPC as given below:

DWORD WINAPI QueueUserAPC(_In_ PAPCFUNCpfnAPC, _In_ HANDLEhThread, _In_ ULONG_PTR dwData);[4]pfnAPC [in]

A pointer to the application-supplied APC function to be calledwhen the specified thread performs an alertable wait operation.For more information, see APCProc.hThread [in]

A handle to the thread. The handle must have the THREAD_SET_CONTEXT access right. For more information, see Synchronization Object Security and Access Rights.dwData [in] A single value that is passed to the APC function pointed to by the pfnAPC parameter [8].Because a thread mustbe in an alertable state in order to run a user-mode APC, malware will look totarget threads in processesthat are likely to go into that state. Luckily for themalware analyst, WaitForSingleObjectExis the most common call in theWindowsAPI, and there are usually many threads in the alertablestate. Let's examine the QueueUserAPC's parameters: pfnAPC,hThread, and dwData. Acall to QueueUserAPCis a request forthe thread whose handle is hThreadto runthe function defined by pfnAPCwith the parameter dwData[8] The malware can useQueueUserAPCto force a DLL to be loaded in the contextof obtained, the malware uses it to opena handle to the thread. In thisexample, the malware is lookingto force the thread to load a DLL in the remote process, so you see a call toQueueUserAPCwiththe pfnAPCset to LoadLibraryA.Once this APC is

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

queued and the threadgoes into an alertable state, LoadLibraryAwill be calledby the remote thread, causing the target process to load dbnet.dll.[9]

```
push    [esp+4+dwThreadId]      ; dwThreadId
push    0                       ; bInheritHandle
push    10h                     ; dwDesiredAccess
call    ds:OpenThread ❶
mov     esi, eax
test    esi, esi
jz      short loc_401DCE
push    [esp+4+dwData]          ; dwData = dbnet.dll
push    esi                     ; hThread
push    ds:LoadLibraryA ❷       ; pfnAPC
call    ds:QueueUserAPC
```

iii. ***APC Injection from Kernel Space :***Malware drivers and rootkits often wish to execute code in userspace, butthere is no easy way for them to do it. One method they use is to performAPC injection from kernel space to get their code execution in user space. A malicious driver can buildan APC and dispatch athread to execute it in auser-mode process (most often svchost.exe). APCs of this type often consistofshellcode.Device drivers use two major functions in order toutilize PCs:KeInitializeApcandKeInsertQueueApc. . The APCfirst must be initialized with a call to KeInitializeApc. If thesixthparameter (NormalRoutine) is non-zero in combination with theseventhparameter (ApcMode) being set to 1, then we are lookingat a usermodetype[1].Therefore, focusing on these two parameters can tell you if therootkit is using APC injection to run code inuser space.KeInitializeAPCinitializes a KAPC structure, whichmust be passed toKeInsertQueueApcto place the APC object inthe target thread's correspondingAPC queue [2].

```
000119BD        push    ebx
000119BE        push    1 ❶
000119C0        push    [ebp+arg_4] ❷
000119C3        push    ebx
000119C4        push    offset sub_11964
000119C9        push    2
000119CB        push    [ebp+arg_0] ❸
000119CE        push    esi
000119CF        call    ds:KeInitializeApc
000119D5        cmp     edi, ebx
000119D7        jz      short loc_119EA
000119D9        push    ebx
000119DA        push    [ebp+arg_C]
000119DD        push    [ebp+arg_8]
000119E0        push    esi
000119E1        call    edi             ;KeInsertQueueApc
```

SleepEx, SignalObjectAndWait, MsgWaitForMultipleObjectsEx, WaitForMultipleObjectsEx, or WaitForSingleObjectEx functions. The malware usually looks for any thread that is in an alterable state, and then calls OpenThread and QueueUserAPC to queue an APC to a thread. QueueUserAPC takes three arguments: 1) a handle to the target thread; 2) a pointer to the function that the malware wants to run; 3) and the parameter that is passed to the function pointer. In Figure 8, Amanahe malware first calls OpenThread to acquire a handle of another thread, and then calls QueueUserAPC with LoadLibraryA as the function pointer to inject its malicious DLL into another thread.

AtomBombing is a technique that was first introduced by enSiloresearch, and then used in DridexV4. As the technique also relies on APC injection. However, it uses atom tables for writing into memory of another process.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

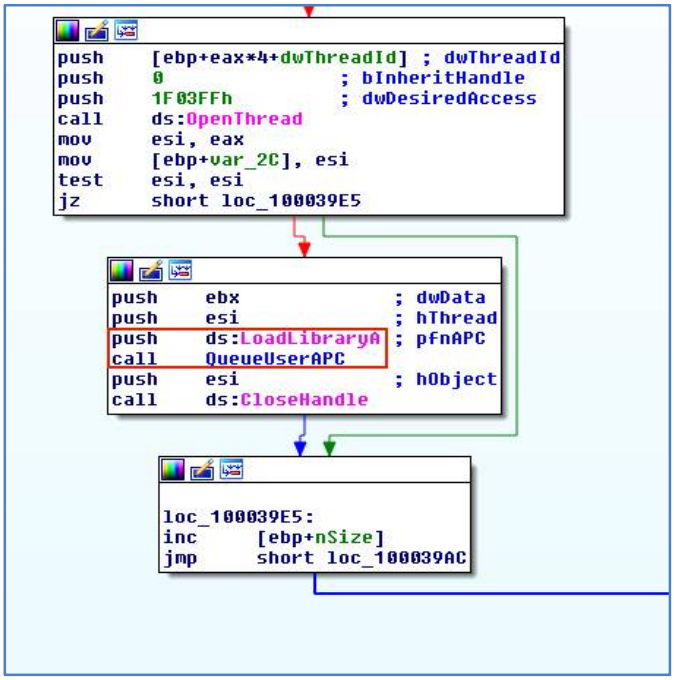
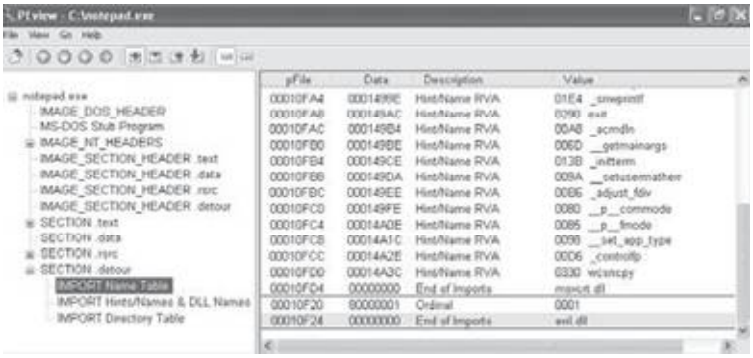


Figure showing *Almanahe performing APC injection*
Sha256: f74399cc0be275376dad23151e3d0c2e2a1c966e6db6a695a05ec1a30551c0ad

G. Detours: Detours is a library developed by Microsoft Researchers [4].It was originallyintended as a way to easily instrument and extend existing OS and applicationfunctionality. The Detours library makes it possible for a developer tomake application modifications simply. Malware authors like Detours, too, and theyuse the Detours library toperform import table modification, attach DLLs to existing program files, and add function hooks to runningprocesses. Malware authors most commonly use Detours to addnew DLLs to existingbinaries on disk. The malware modifies thePE structure and creates asection named .detour, which is typically placed between the export table andany debug symbols [5]. The.detour section contains the original PE header witha new importaddress table. The malware author then uses Detours to modifythe PE header to point to the new import table, by using the setdlltoolprovidedwith the Detours library. Instead of using the official Microsoft Detours library, malware authorshave been known touse alternative and custom methods to add a .detoursection. Theuse of these methods for detour addition should not impactyourability to analyse the malware [6].



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 6, June 2017

H. Alternate Data Streams: Alternate Data Stream (ADS) is the lesser known feature of Windows NTFS file system which provides the ability to put datainto existing files and folders without affecting their functionalityand size. Any such stream associated with file/folder is not visiblewhen viewed through conventional utilities such as Windows Explorer or DIR command or any other file browser tools. It isused legitimately by Windows and other applications to store additional information (for example summary information) forthe file. Even 'Internet Explorer' adds the stream downloaded from the internet. Due to this hidden nature of ADS, hackers have been exploitingthis method to secretly store their Rootkit components on the compromised system without being detected. This makes the stream a hidden threat [1].

III. CONCLUSION

In this paper we have analysed common covert methods through whichmalware launches, ranging from the simple to advanced. Many ofthe techniques involve manipulating live memory on the system, as with DLL injection, process replacement, and hook injection.Other techniques involve modifying binaries on disk, as in the case of adding a .detour section to a PE file. Although these techniquesare all very different, they achieve the same goal.A malware analyst must be able to recognize launching techniques in order to know how to find malware on a live system. We understand that how malware Authors hide malware from detecting the malicious activities of malware from detection toolsusing various covert launching technique [1].

REFERENCES

[1] Static Analysis Based Behavioral API forMalware Detection using Markov Chain Proff. Abbas M. Al-Bakri1 Hussein L. Hussein2 1.Kuffa university/Faculty collage of computer and mathe-matic 12, 2014
[2] I Read<http://files-recovery.blogspot.co.in/2011/04/27-malware-types-and-their.html>27 Mal-ware Types and Their Characteristics<http://files-recovery.blogspot.co.uk/2011/04/27-malware-types-
[3] Brian MARINI, High-Tech Bridge,"Information Security Solutions", 6th Sep.2011.
[4] Grecs,"Malware Analysis:Noob to Ninja in 60 Minutes".
[5] Chris Eagle,"The IDA Pro Book", 2nd Edition.
[6] Rannoh/Matsnu,"Malware Analysis".
[7] Brent E. Rector, Joseph M. Newcomer,"Win32 Programming (Addison-Wesley Advanced Windows Series)(2 Vol set)".
[8] [Online] Available: http://www.rand0m5ec.blogspot.in/2013/03/dll-code-injection.html
[9] [Online] Available: http://www.coderbag.com/Threading/DLL-Injection-Using-Remote-Thread

BIOGRAPHY

SurabhiDubey (Author): She received B.Tech degree in Computer from MIETGondia (NagpurUniversity).Currently pursuing M.Tech from MDU University Rohtak -Haryana, India and working as Independent Security Re-searcher Security analysis.

Mrs.Anita Tomer(Co-Author): Currently H.O.D**.** Computer Science&Engineering, Department of Computer Science & Engineering Delhi Institute of Technology, Management & Research Faridabad affiliated to MaharshiDayanand University, Rohtak- Haryana (India)