www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume – 5 Issue -03 March, 2016 Page No. 16038-16040

Advance Dynamic Malware Analysis Using Api Hooking

Ajay Kumar, Shubham Goyal
Department of computer science
Shivaji College, University of Delhi, Delhi, India
ajay.cs@shivaji.du.ac.in imshubhamgoyal@gmail.com

Abstract— As in real world, in virtual world also there are people who want to take advantage of you by exploiting you whether it would be your money, your status or your personal information etc. MALWARE helps these people accomplishing their goals. The security of modern computer systems depends on the ability by the users to keep software, OS and antivirus products up-to-date. To protect legitimate users from these threats, I made a tool
(ADVANCE DYNAMIC MALWARE ANAYSIS USING API HOOKING) that will inform you about every task that software (malware) is doing over your machine at run-time

***Index Terms*— API Hooking, Hooking, DLL injection, Detour**

I. INTRODUCTION

Malwares are most commonly used in security breach and plays an important part in security incidents. Malware Analysis is an art used to dissect any malware to understand how it works, how to identify it and how to eliminate it. Sometimes we are provided with only malware executable, which won't be human-readable. In order to make sense of it, Malware Analyst has to use a variety of tools and tricks, each revealing a small amount of information.

II. OBJECTIVE

My aim is to perform malware analysis on any windows based software during runtime. Many of the programmers use API calls to access system resources like files, processes, changing registry files etc. We can check what these malware/software(s) are doing by monitoring the relevant APIs and their parameters. I have used API hooking for this.

III. MALWARE ANALYSIS

Malware Analysis is the study of a malware by dissecting its different components and studying its behavior on the host computer's operating system. Two types of analysis could be done:

A. Static Malware Analysis

Analyzing software without executing it, this is usually done by dissecting the different resources of the binary file and studying each component. The binary file can also be disassembled using a disassembler.
Basic static analysis consists of examining the executable file without viewing the actual instructions. Basic static analysis can confirm whether a file is malicious, provide information about its functionality, and sometimes provide information that will allow you to produce simple network signatures. But this type of Analysis is ineffective against many sophisticated software. Advanced static analysis consists of reverse-engineering the malware's internals by loading the executable into a disassembler and looking at the program instructions in order to discover what the program does. Advanced static analysis tells you exactly what the program does.

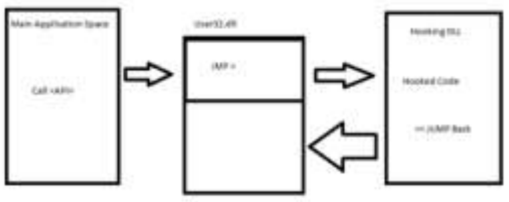
B. Dynamic Malware Analysis

This is done by watching and monitoring the behavior of the malware while running on the host. Virtual machines and Sandboxes are extensively used for this type of analysis. The malware is debugged while running using a debugger to watch the behavior of the malware step by step while its instructions are being processed by the processor and their live effects on RAM. This technique provides very fast and accurate way to extract detailed information from an executable.

One possibility to monitor what functions are called by a program is to intercept these calls. The process of intercepting function calls is called ***hooking***. *The analyzed program is manipulated in a way that in addition to the intended function, a so-called hook function is invoked*. This hook function is responsible for implementing the required analysis functionality

IV. HOOKING

API hooking is a technique by which we can instrument and modify the behavior and flow of API calls.



A. IAT Hooking

Import Address Table (IAT) is an array of links representing the various DLLs imported by the PE loader during process initiation. IAT hooking is a technique of modifying the address of a particular DLL in the IAT with address of hook function. Before performing IAT hooking we must make sure that we are able to put the hook function in the user's address space through any of the DLL injection methods

B. Inline Hooking

An inline function hook is implemented by overwriting the beginning of target function with an unconditional jump to a

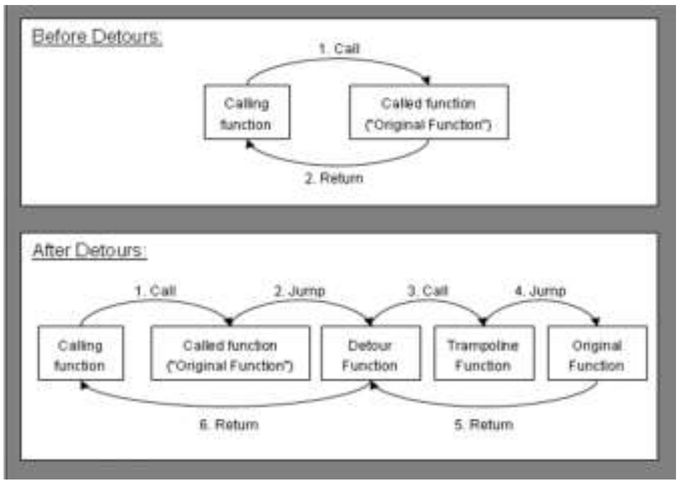
Detour function. Detour function calls a Trampoline function, which contains the overwritten bytes of the original target function, and then calls the target function. The target function returns to the detour function which finally gives control back to the source function.

V. DLL INJECTION

DLL injection is a technique used for running code within the address space of another process by forcing it to load a DLL. DLL injection is used by almost every malware to place malicious routines in user memory. Though DLL Injection will just place a DLL in memory, executing code present in the DLL is triggered after API hooking is done.

VI. DETOUR

Microsoft released a framework to help in placing inline hooks on Win32 API functions called Detours. Microsoft actually places a two-byte dummy instruction that does nothing at the start of functions (the instruction is “MOV EDI, EDI”) to allow space to overwrite with a jump instruction harmlessly. The Detours package provides an API to enable custom hooks to be placed on API functions in this way



API hooking consists of intercepting a function call in a program and redirecting it to another function. By doing this, the parameters can be modified , the original program can be tricked if you choose to return an error code when really it should be successful, and so on. All of this is done before the real function is called, and in the end, after modifying/storing/extending the original function/parameters, control is handed back over to the original function until it is called again.

The Microsoft Detours library is a library for intercepting arbitrary Win32 binary functions on x86 machines. Interception code is applied dynamically at runtime. Detours replaces the first few instructions of the target function with an unconditional jump to the user-provided detour function. Instructions from the target function are placed in a trampoline. The address of the trampoline is placed in a target pointer. The detour function can either replace the target function, or extend its semantics by invoking the target function as a subroutine through the target pointer to the trampoline.

The function that drives all of this is the DetourAttach(…) function.

LONG DetourAttach(PVOID * ppPointer,PVOID pDetour);

This is the function that is responsible for hooking the target API. The first parameter is a pointer to a pointer of the function that is to be detoured. The second one is a pointer to the function that will act as the detour. However, before the detouring begins, there are a few things that need to be done:

A detour transaction needs to be initiated.

A thread needs to be updated with the transaction.

This is easily done with the following calls:

DetourTransactionBegin()

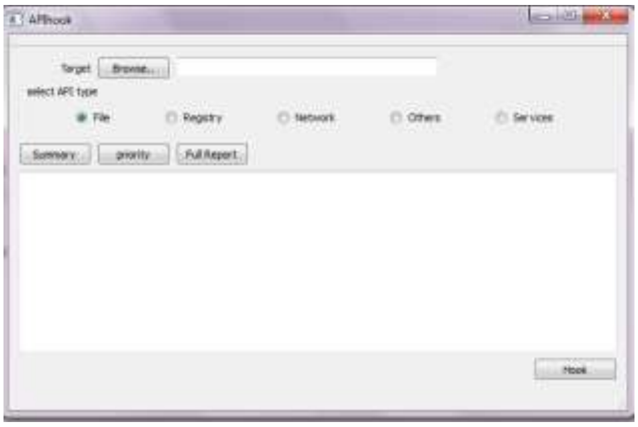
DetourUpdateThread(GetCurrentThread())

After these two things are done, the detour is ready to be attached. After attaching, it is important to call DetourTransactionCommit() to make the detour go into effect and check for success or failure, if need be.

VII. CONCLUSION

At last I made a tool, to intercept any executable and generate text files containing list of all API’s and their parameters used in that executable and changes made by it.

This API hooking tool is a GUI tool that has been made using QT framework which is an open source framework to create GUI. The DLL and The injection code is made in C language.



A. WORKING

Browse Button : Browse button is used to fetch the target application or target malware which we want to analyse and check which WINAPIs it is calling.

Hook Button: Once the target Application is fetched then to hook the detour dll to the the application we will click on hook

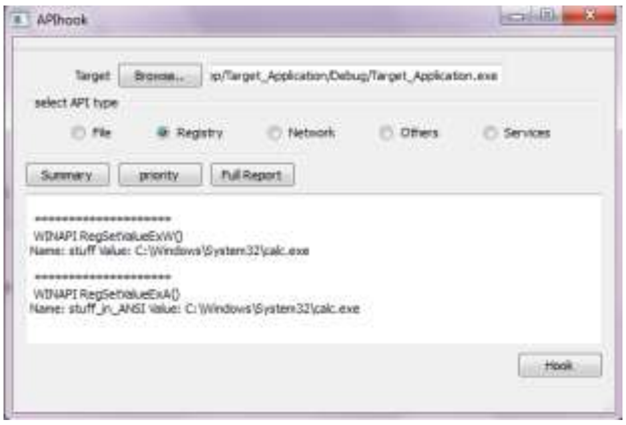
On Clicking Hook Button, DLL named detourdll.dll is injected into the memory space of the target application.

A safe option when finalizing your figures is to strip out the fonts before you save the files, creating “outline” type. This converts fonts to artwork what will appear uniformly on any screen.

Now as soon as the target application call for example send( ) WINAPI the DLL receives the function call and output the data buffer that was passed to send( ) API and

DOI: 10.18535/ijecs/v5i3.32

then returns the same argument to the actual send( ) and makes a function call thus allowing the target application to perform its function



Target: p/Target_Application/Debug/Target_Application.exe

select API type: File [✓ Registry] Network Others Services

```
====================
 WINAPI RegSetValueExW()
Name: stuff Value: C:\Windows\System32\calc.exe

====================
 WINAPI RegSetValueExA()
Name: stuff_in_ANSI Value: C:\Windows\System32\calc.exe
```

We have divided the APIs into five categories:
Network APIs
File APIs
Service APIs
Process APIs
Other APIs

After the hooking is completed the output of the tool is stored in five different directories named as per different categories which contains the information about the function call relation to that API type for example if target application makes send( ) function call the details about the argument passed are stored in network directory.

Each directory contains three text documents
priority.txt : contains only the priority information about the system call made.
summary.txt : contains summary about the function calls.
fullreport.txt: contains all the information about the function calls.

API Type Radio button : We can select the type of API we want to see the information about using this radio button

Summary/Priority/FullReport button: After selecting the type of API on which we want to view the information we will go ahead with selecting the type of report we want to view that we want to view i.e. priority report, summary report of full report

Once we select the report type it will be displayed in the text area we can change the API type and report type as per whatever we need to see.

References

[1] Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software By Michael Sikorski, Andrew Honig, Publisher: No Starch Press 2012 ,800 Pages, ISBN: 1593272901

[2] M. Egele et al, “Survey on Automated Dynamic Malware Analysis Techniques and Tools”, ACM Computing Surveys, Vol. V, No. N, 20YY.

[3] C Willems, T Holz, F Freiling, “Toward automated dynamic malware analysis using cwsandbox”, IEEE Security & Privacy, 32-39, 2007.

[4] Galen Hunt and Doug Brubacher, Detours: Binary Interception of Win32 Functions, in Third USENIX Windows NT Symposium, USENIX, July 1999.