Dynamic Analysis of a Suspected Stuxnet Malicious Code

Wang Guangwei*, Pan Hong**, Fan Mingyu***
School of Computer Science and Engineering
UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA
Chengdu 611731
*wgw1339@163.com
**phthegreat@139.com
***ff98@163.com

***Abstract*—Stuxnet known as "shock" is a malicious code for Microsoft systems and Siemens industrial systems.It is sweeping the globe.Industrial systems and individual users in many countries and regions are infected. Different from previous malicious code, the code is very complicated, but less relevant analytical experiments reported. Weanalyzea newly discovered malicious codeby dynamic analysis experiments. The result shows that the sample isvery similar to Stuxnetin functional characteristics.**

Keywords-malicious code; dynamic analysis; shock

I. INTRODUCTION

Stuxnet[1],[2],[3], also known as"shock" virus, is a malicious code which has infected more than 45,000 networks worldwide, Iran suffered the most serious attack, 60% of the personal computers were infected4. It is reported that nearly 500 million Chines Internet users, and various leading companiessuffered the attack. So far the analysis in publicly reported isless.This paper presents a dynamic analysis process on newly discovered malicious code. Experimentalresult shows that the sample is very similar toStuxnet.

Analysis of the experimental environment is CPU 2.20GHz; 2G; operating system Windows XP.

Testing tools are PEID[5], OD[6], and IDA[7], where PEID is for shell analysis,OllyDbg (OD) is for dynamic tracking and IDA is for static analysis.

Sample is a newly discovered malicious code.

The Objective is analysis the sample, findingits malicious behavior.

Because of the limited length, this paper will not give all the evidence in the analysis process, only gives the main flow in dynamic tracking.

II. THE PROCESSES OF EXPERIMENTAL ANALYSIS

Analysis of experiments has three steps:

Packeranalysis: the objective is to examine whether the sample has packers protection.

Behavior analysis: dynamic tracking combined with static analysis, to obtain the behavior process of the sample, to determine possible malicious behavior

Summary of malicious behavior analysis: giving malicious behavior of the sample according to the analysis results.

III. THE EXPERIMENTAL ANALYSIS

A. Packer Analysis

Packer analysis found no packers. Its export table information (Figure 1) shows that it belongs to a control plane programmed.

B. Behavior AnalysisExperiment

a) The main flow

The sample Dynamic tracked, it runs into the second function, DllCanUnloadNow. In this function, the sample determines the operating system version. When System platform is NT, and the major version number must be 5 and 6, that is the operating system is WIN XP/VISTA/7, the sample runs into the main flow.

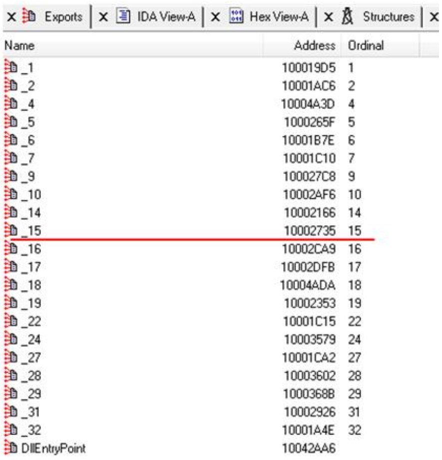
In the main process, the sample mainly completes: decryptthe resource file resource.dll. After establishing the section, memory mapping, the decrypted sample resource file is loaded into memory as a module.From the overall point of view, the sample rewrite the system API LoadLibrary.

After analysis,the resource file is located at offset 0x622c, size 0x79a00, and it is a dll file. After decryption, it calls functions such as ZwCreateSection, ZwMapViewOfFile.Then it maps the dll file into the application section. During operation it also HOOKsZw family functions such as ZwMapViewOfSection, ZwOpenFile, ZwClose, ZwQueryAttributeFile, ZwQuerySection, ZwCreateSection.Its aim should be to cheat the system, causing that the dll file is saved in the hard disk rather than in memory. Then, loading the dll file is logical.

After successfully load the dll file, it calls function No. 15exported from dll, then control is passed to the dll. The following analysis is transferred to the dll file.

b) Resource File

Get the dll file from the memory dump using OD. Packer analysis shows that it waspacked. After unpacking, the dll's import table is analyzed, a number of exported functions are found.As shown in Figure 1.



Name	Address	Ordinal
_1	100019D5	1
_2	10001AC6	2
_4	10004A3D	4
_5	1000265F	5
_6	10001B7E	6
_7	10001C10	7
_9	100027C8	9
_10	10002AF6	10
_14	10002166	14
_15	10002735	15
_16	10002CA9	16
_17	10002DFB	17
_18	10004ADA	18
_19	10002353	19
_22	10001C15	22
_24	10003579	24
_27	10001CA2	27
_28	10003602	28
_29	1000368B	29
_31	10002926	31
_32	10001A4E	32
DllEntryPoint	10042AA6	

Figure 1 exported functions from resource file

As the sample calls No. 15 function exported from it, therefore the first analysis is focused on the 15th exported function.

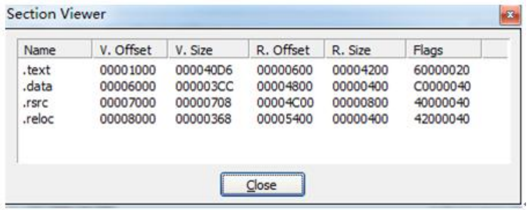
c) 15th function

We enter into the main flow of the 15th function.

First, it determines whether the current user is an administrator, if yes the administrator's session token is copied. Then it traverses and analyzes the process to determine whether the system has the following process, avp.exe, McShield.exe, avguard.exe, bdagent.exe, UmxCfg.exe, fsdfwd.exe, rtvscan.exe, ccSvcHst.exe, ekrn.exe, tmproxy.exe. If there is one, it reads the location of the found files from the registry, and saves the file information. Then add the environment variable "% SystemRoot%\\system32\\lsass.exe". Finally, the administrator session token is used to start lsass.exe hanging.

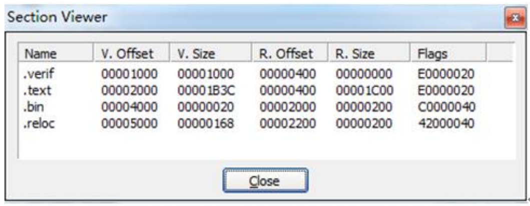
After Lsass.exe is hang, the dll fileuses functions such asZwCreateSection, ZwMapViewOfSectionas before, and adds itself to lsass.exe memory space, and then resume the main thread. After Lsass.exe startup, it will load resource.dll. So it can be judged that the 15th function is to obtain information of specified process, and then inject itself into the lsass.exe process.

Figure 2 is the original section information in lsass.exe. Figure 3 is the modified section information in lsass.exe.



Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.text	00001000	000040D6	00000600	00004200	60000020
.data	00006000	000003CC	00004800	00000400	C0000040
.rsrc	00007000	00000708	00004C00	00000800	40000040
.reloc	00008000	00000368	00005400	00000400	42000040

Figure 2. The original section information in lsass.exe



Name	V. Offset	V. Size	R. Offset	R. Size	Flags
.verif	00001000	00001000	00000400	00000000	E0000020
.text	00002000	00001B3C	00000400	00001C00	E0000020
.bin	00004000	00000020	00002000	00000200	C0000040
.reloc	00005000	00000168	00002200	00000200	42000040

Figure 3. The modified section information in lsass.exe

We can find that, After modification, PE head is completely changed. Analysis the modified lsass.exe, we can find that the loaded lsass.exe start a new thread. In the new thread, loading resource.dll is used, that is, Hook several functions such as ZwMapViewOfSection, ZwCreateSection, ZwOpenFile, ZwClose, ZwQuery Attributes Files, ZwQuerySection, resulting in the illusion of the resource.dll copy of lsass.exe memory existing in hard disk, and then when you call Load Library, resouce.dll will be loaded into lsass's module list.

After loading is complete, it reads relevant data of the samples mapped, these data contain the memory starting address of the copy of resouce.dll, the function number which will be called, and then start a new thread again, calling resource.dll exported 16 functions.

So far, lsass.exe modified work completed. As can be seen from the results reflected, the modified lsass.exe plays the role of a puppet process. From outward appearances it is a system critical process, but memory code has been replaced, its purpose should be to deceive antivirus software.

The following analysis will be the 16th function exported from resource.dll.

d) 16th function

In the 16th function, it first generates three temporary file named "~ DF + random number" in the user temporary directory.

The three documents areciphertext forms of resource.dll, and related configuration data. Next, we check the registry data, the registry key is:

“HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\MS-DOS Emulation\NTVDM TRACE”

Read the key contents to determine whether it is equal to "0x19790509", if they are equal, then end the work, otherwise continue working.

Create a global mutex variable named "Global \ {62BBECCC-536F-4dc6-A387-8B1A17CF8A75}".

Check the file creation time, if the time is later than June 24, 2012, then end the work, otherwise continue working.Therefore, we need to adjust the system time in the analysis process.

If passed the time test, then it releasestwo drivers named "Mrxnet.sys"and "Mrxcls.sys "in the drive directory, two drivers are registered as a system service, with the following registry key:

“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRxCls”

“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MrxNet”

After releasing of the two documents, it communicates with control terminal, testing whether the services installation is successful.If successful, thenit activatesthe global mutex variable, otherwise continues services installation. Then reading the registry keys to determine whether have: “ HKEY_LOCAL_MACHINE SOFTWARE\SIEMENS\STEP7\ STEP7_Version”

If there is no then end the work, otherwise continue working.

From these registry paths we may know, this is the Siemens PLC programming software.Because the track was not able to perform, we cannot know the process further.

The following analysis is transferred to two drivers releasedfrom 16th function.

e) Two drivers

Driver MrxCls is set to boot drive,this means that after the system initialization is complete, the drive is automatically loaded. MrcCls registered callback function to start the process. It observes that whetherthe followingfour processes are started:

“services.exe”,

“S7tgtopx.exe”,

“CCProjectMgr.exe”

“explorer.exe”

If it finds these processes starts, it will inject relevant code of resource.dll into them.

From thedriven function, we can see that this is a self-start way of the sample.With the opportunity and privileges of driver to start and run, the resouce.dll is injected into the boot process, and randomlystarts.

MrxNet is a device driver, it loads itself into the device stack of the file system. This means MrxNet can intercept any file operation request packages generated by the system, including read, write, delete. In addition, MrxNet also registered a device to increase the callback, the callback observes whether it is a mobile device.

From these operations of MrxNetwe can guess, MrxNetwill mainly infect files on mobile device. When a mobile device is plugged, MrxNet will write resource.dll to the mobile device. When some files are modifed in the mobile device,MrxNetwill hijack file operations, and will write resource.dll as data in the file.

From the driven function, we can see that the infection way of the sample isto hijack system file operations with drive. It observes the access of the new device, once it finds file operations, it will write resource.dll to the mobile device.

f) Summary of malicious behavior of the sample

The sample contains a dynamic link library resource.dll, the main function of the sample is done inside the dll.This dllwill derive 21 functions, among them the 15th function is startup suspend as a system process lsass, then maps resource.dll and related data to the lsass process, and modifies the PE header of lsass. After modification, lsass code in memory all is replaced. Then the function calls on the 16th to complete the self-starting and infection.

The sample distributes different functions of resource.dll in different processes, this brings a lot more difficult to analyze. The Sample makes an extensive use of memory mapping that injects code into the process;its purpose is to escape the anti-virus software.The sample is highly targeted for Siemens Step7 PLC programming software. The sample also uses drive technology to achieve self-starting and infecting other files.The sample has a limited time, when the time is later than June 24, 2012, it will not work.

IV. EXPERIMENTAL ANALYSIS RESULTS

Experimental results show that, this sample is a malware. Its implementation is very complex.The resource.dll is distributes to different functions in different processes, specifically for Siemens PLC programming software Step7. It extensively uses adding section of memory and mapping memory files to load code into files.It is very similar to the well-known Stuxnetin functional characteristics, it is a sophisticated malware.

REFERENCES

[1]Stuxnet, http://en.wikipedia.org/wiki/Stuxnet

[2]Michael Kelley, The Stuxnet Attack On Iran's Nuclear Plant Was 'Far More Dangerous' Than Previously Thought, http://www.businessinsider.com/stuxnet-was-far-more-dangerous-than-previous-thought-2013-11, Nov. 20, 2013

[3]Ralph Langner: Cracking Stuxnet, a 21st-century cyber weapon, http://www.ted.com/talks/ralph_langner_cracking_stuxnet_a_21st_century_cyberweapon

[4] David Kushner, The Real Story of Stuxnet, http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet, Feb 2013

[5] PEiD, http://www.softpedia.com/get/Programming/Packers-Crypters-Protectors/PEiD-updated.shtml, May. 2013

[6]Ollydbg,http://ollydbg.net/, May2013

[7]IDA, https://www.hex-rays.com/products/ida/index.shtml, May 2013