Identifying The Malware Persistence Using Advance Static And Advance Dynamic Method

Lakshaya Dubey

Abstract: As the covert nature of the malware has increase it have become easier for the malware to bypass the security system. Using the various intrusion mechanism the malwares manage to gain the persistence over the target system. The malware are the most sophisticated evil code which is designed to harm the system without the knowledge of the owner of the system. Malware often uses persistence so that the malware author can communicate to the affected system even after the system gets reboot or log-off. Using the persistence on the system malware author can also use the effected system to exploit the other system in the local network or in the remote location. For identifying the malware persistence we will use the process of malware analysis. Malware analysis is the process of analyzing the malware of identify the nature of the malware. In the research will use the combination of static analysis, dynamic analysis and the advance static analysis and advance dynamic analysis technique which will help us to find the various persistence mechanism used by the malware.

Index Terms: Covert, Dynamic analysis, Malware, Persistence, Privilege, Static analysis.

1. INTRODUCTION

Malware authors often spends a lot of time in writing the malware and finding a way to make them sly. They even find the unique way to make them run on the target machine. The persistence mechanisms make sure the backdoor remain installed and running even after the system reboots. Starting with reconnaissance of the target, installation of malicious code, establishing a command and control(C&C) channel, and accomplishing the mission-which is often the process of exhilarating data is the common attack flow. In this process the persistence plays an important role. Having the persistence over the system can also allow the attacker to gain access to the target connection remotely. This remote connection can later be used for the further exploitation of the other targets and its infrastructure. The complexity of the persistence method used by the attacker usually depends on the privileges he have over the target machine. Higher the privileges the more sophisticated and stealthy persistence can be applied. There are multiple approaches of delivering the malware to the target system, like spear phishing email, social engineering, malicious email attachments or exploits which uses the vulnerability present over the target system. While there many other approaches for delivering the malware over the target system, the end result is the same, attacker gain control of the victim's machine.The malware delivering process might be very complex in a way that multiple stages of malicious code are executed. This is mainly done due to specifics of the malicious payload delivery or in order to bypass security defenses which might detect the initial compromisation. In this research Malware sample, ABC.exe, LSD.exe Pbkdf.exe and RSA.exe will be analyzed using advance static and advance dynamic to identify their persistence mechanism.

2 MALWARE PERSISTENCE TECHNIQUES

2.1 Bootkit/Rootkit

Rootkit/Bootkit is a collection of malicious program that can modify startup code like the Master Boot Record (MRB), Volume Boot Record (VBR) and a Boot sector. The malware targets MRB, VBR, or a boot sector located on the physical motherboard of the computer. Attaching malicious software in this manner which can allow the malicious program to be executed prior to the loading of the operating system.The advantage/disadvantage of the bootkit /rootkit infection is that it cannot be detected by the standard AV's (anti-virus) and OS, because all of the components reside outside the windows file system. Due to this qualities of bootkit /rootkit they have become advantageous for the attackers and disadvantageous for the target system. The rootkit/bootkit has three goals:

I. Run: For the functioning of the malware, it has to be executed on the target machine for that the malware author can exploit the vulnerability over the target system and install the backdoor as a persistence.
II. Act: A rootkit/bootkit has a specific action, which the author wants. Running in the covert mode is well and good, but there is
III. extra the author wants like stealing passwords, exploiting other machines and it's infrastructure.

The rootkit/bootkit destabilize the normal OS behavior. This defile occurs when a rootkit hides by lying to other software on the computer. Mostly all the software relies on the OS to provide the information about the Environment in which it is running and mostly all Applications have dependencies (dlls, registry etc.).The application asks for those dependencies using API (Application Program Interface) Provided by the operating system, then the operating system return the appropriate information to the application. This same APIs are often used by the security software when scanning for the malicious program by the anti-virus programs. In order to hide, the rootkits hijack these APIs and watch for any query made by the AVs in search of file, the rootkit not only prevent the file and also deny the existence of that file. Techniques a rootkit/bootkit can use to defile the normal OS behavior:

1) Hijacking the OS APIs
2) Hiding in the empty and unused space on the machine's HDD (hard disk drive)
3) Infecting the MRB (master Boot Record)

- *Lakshya Dubey is currently pursuing bachelor degree program in computer science engineering in Rajiv Gandhi Proudyogiki Vishwavidyalaya, India, PH-9407178518, E-mail: sshlakshya@gmail.com*

Infection Cycle

Once a rootkit infector has been downloaded on the target system through one of its disposition channels, it begins the infection process. In order to survive a system reboot, rootkit infects one of the boot-start drivers essential to

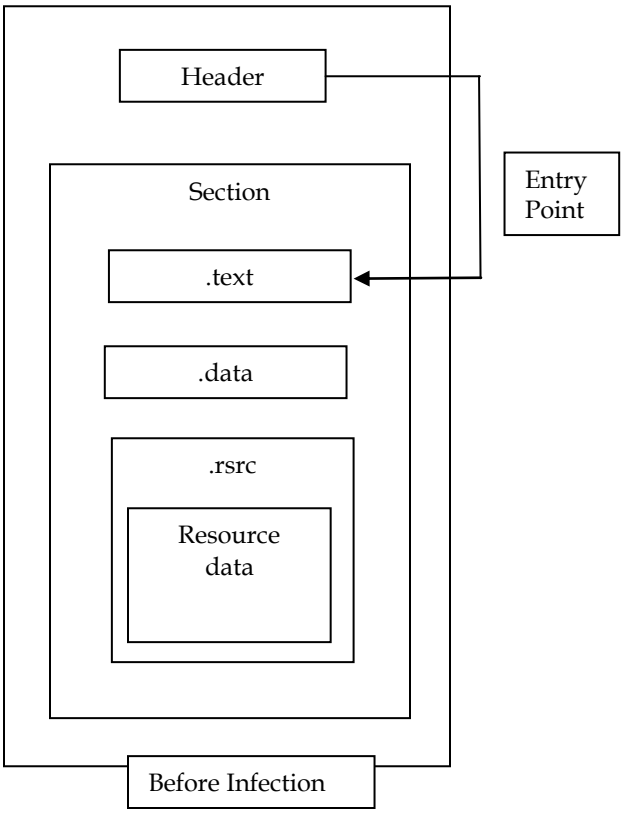

Fig. 1*. Kernel-mode boot-start driver before infection of the system.*

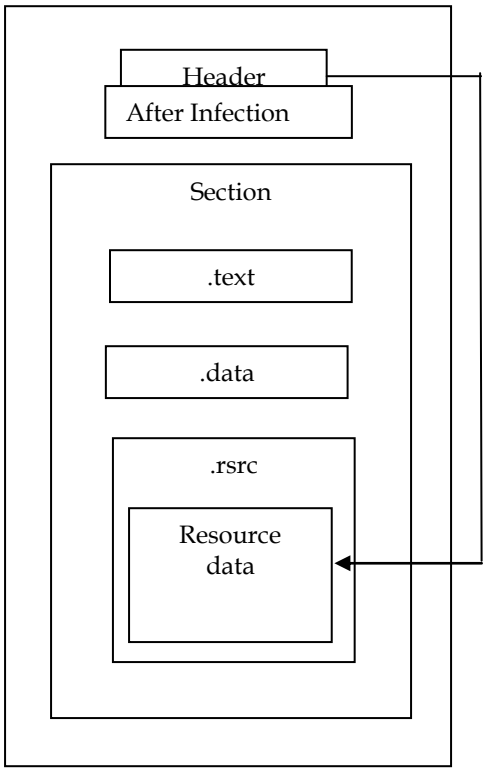

Fig. 2. *Modification of kernel-mode boot-start driver upon infection of the system.*

2.2 Modifying registry keys

Registries are often modified by the malware to gain the persistence on the system by the malware. As windows has many AutoStart Extension Points (ASEP), so most malware achieve the persistence by editing them. The common registry keys modified by the malware are:

The keys used by the malware to gain persistence at user level

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices

Entry Point

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce

The keys used by the malware to gain persistence at root/admin level

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

2.3 Application Initialization DLLs (appinit_dlls)

The AppInit_DLL registry key was mostly used in Windows NT and post Windows 95 machines, as AppInit_DLL is the popular attack vector for persistence. It makes every process that executes load USER32.dll.Almost all processes load this DLL for their use, and this makes it the great means of loading malicious code which gives persistence over the target machine.
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs

2.4 Startup folders

This is the simplest way to gain persistence on the target machine for the specific user which does not requires administrative privileges. Because of the cool feature this technique is most common across various attackers. In order to deploy persistence for all users via this technique administrative privileges are required, as the home directory or folder is protected by the OS (operating System). For this the malware have to escalate the privileges after getting the user access over the system. To gain persistence over the target machine using the user, copy of the malware should be present in the home directory of that user. According to Windows XP: C:\Documents and Settings \% USERNAME %\ Start Menu \ Programs \ Startup To gain persistence using the administrator user, the copy of the malware should be present in the home directory of the admin user. According to Windows XP: C:\Documents and Settings\All Users \ Start Menu \ Programs \ Startup

2.5 DLL Search Order Hijacking:

Since many AV's sucks at detecting DDLs, this is the stealthy persistence technique used by malware. As import address tables don't specify the full path, Hijacking abuses the DLL search pattern. Sometimes many programs try to import the DLLs which doesn't even exists on the system. And when the DLL fails to load, the code blindly takes another path, and if the attacker manage to supply our malicious DLLs, the application will blindly loads it for us. The searching of DLLs is started from the directory which contains the EXE. So as the application starts the search of DLL very first starts from the directory containing the EXE. This technique is a great way to get initial code execution on a system but makes poor malware persistence mechanism, because many security vendor recommended loading DLLs by the explicit paths, which is the valid approach.

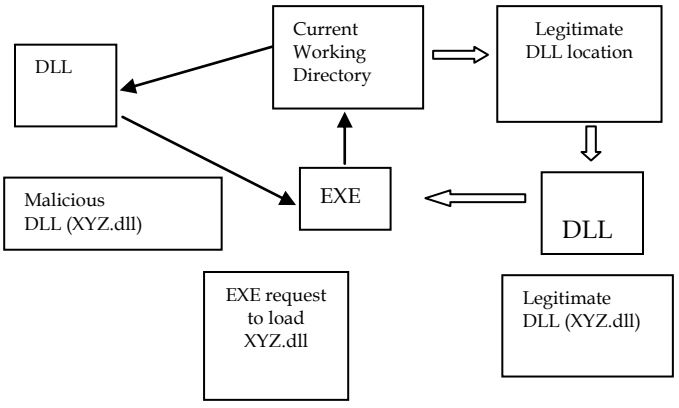

Fig 3: *DLL Search Order Hijacking.*

Since the locations where programs store their DLLs can easily be identified, attackers may place malicious DLLs high up in the path traversed to find the legitimate DLL. Therefore, when Windows searches for a certain DLL in its normal location, it will find a DLL file with the same name but it will not be the legitimate DLL. Often, this type of attack occurs to programs that store DLLs in remote locations, such as in web shares. The DLLs are therefore more exposed to attackers and they no longer need physically to get to a computer so as to compromise files on hard disk drive. The technique is mostly used in the exploits of privilege escalations to carry the further track. Another approach to DLL search order hijacking is the modification of the way in which the program search the DLLs. The attacker can modify the search pattern of the DLLs while they are loading in the program, the attacker can load the malicious DLLs instead of legitimate DLLs to gain the privileges and persistence. In this case the attacker targets the program which are being executed with high level privileges on the system. When done to the right program, the attacker could essentially escalate privileges to become a system user and therefore, have access to more things.

2.6 Use of Mutex Object

Malicious Program use mutex object to defend the AVs and also to gain the persistence on the target host. Legitimate software use mutex object as a locking mechanism to serialize access to a resource on the system, similarly malicious software often uses mutex object for the same purpose as legitimate software. Mutex is not used by all the malware, but if the malware is using it then the mutex object can act as a persistence for that malware. Mutex object can help the malware developer to connect to the target system. As the malware infects a system, the first step is to obtain a handle to a named mutex, if the process fails then the malware exits. The simplest and the easiest way to check for the presence of mutex in the program is debugging the malware, the malware must contain CreateMutex Function. This is the same function that malware uses for checking if the system is infected so the easiest way to identify that the host is infected by the malware is by trying to create the mutex object, if the mutex object is already present then the malware will exit or it will create the mutex. When responding to the infected system, we can analyze the known malware to identify that is the malware is using the mutex object and after identifying the mutex we can also analyze that mutex object of identify its persistence mechanism and its behavior. Some complex and sophisticated malware use the mutex object for serialization or running the single copy of the malware over the target system.

2.7 Process Injection/Process Hollowing

Process Injection or Process Hollowing as the name suggest, injecting some bits of code in the running process, more precisely injecting the executable section of a legitimate process in the memory is replaced with the malicious code or application. Using this technique the malware authors can use legitimate process to execute there malware. The advantage of using this technique is that the path of the process being hallowed out will still point to the legitimate path and by executing within the context of legitimate process the malware

can by-pass the security protection system like HIDS and firewall. As the memory of the executable section of the legitimate process is replaced by the malicious code, this allows the attacker to remain undetected while investigating with live forensic tools. The malicious code can be of any type DLLs, EXE etc. And as the malware or the malicious code is running with the legitimate program the security products will not stop the malware from acquiring the persistence. There are various types of Process injection:

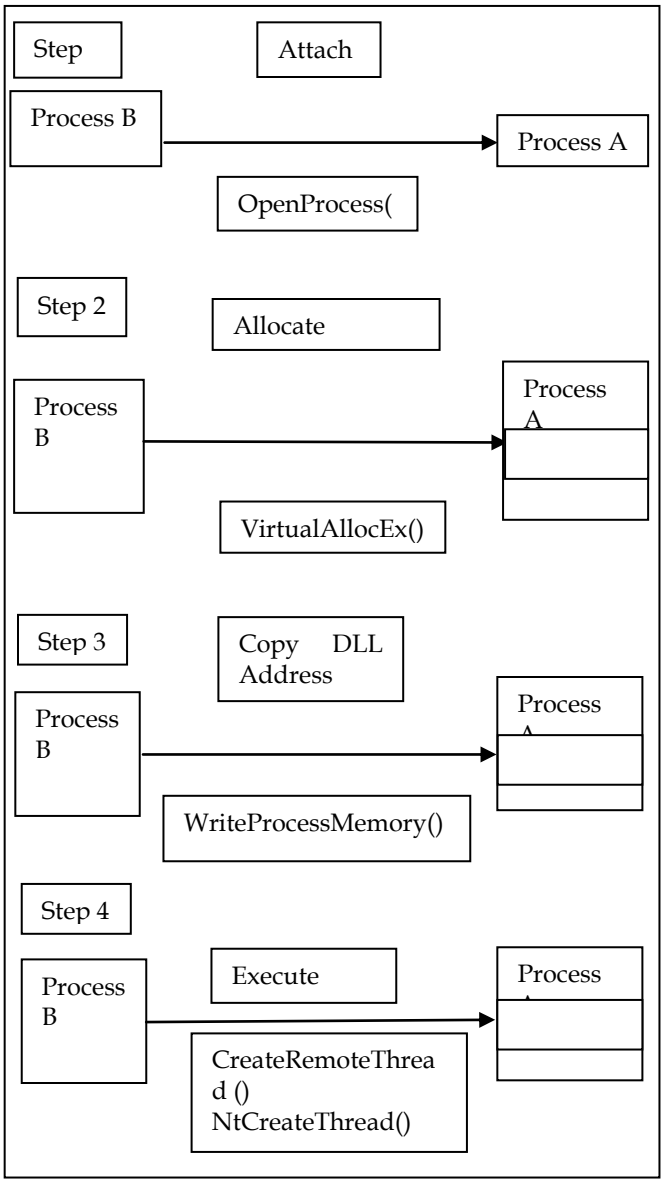

Fig 4: *DLL Injection.*

- Dynamic Link Library injection (DLL-Injection):- The type of process injection in which we inject the DLL into the running process. As the process runs it requests various DLLs, the attacker can change the path of legitimate DLL with the malicious DLL. And as the program invoke the malicious DLL the malware will manage to gain persistence over the system.
- PE Injection:- The Portable Executable format is the standard file format used by windows executable, DLLs and object code in both x86 and x64 bit architecture of windows system. The PE format encapsulates all the necessary information for the management of that encapsulated executable code. This technique doesn't rely on the LoadLibrary() function, this function helps the PE files to load the library from the disk. This technique works by creating a DLL that maps itself into memory When executed, instead of relying on the windows loader, which copies its malicious code into an existing open process and cause it to execute by using a shell code, or by calling the CreateRemoteThread() function. And while executing malware allocates memory in a legitimate host process by using the WriteProcessMemory() to write its malicious code in the legitimate host process.

Infection Cycle of PE injection:

- Fetch the details from the PE header like current image base address and size of PE header.
- Use VirtualAllocEx() to allocate enough memory for the image inside the processes own address space.
- Use VirtualAllocEx() to allocate enough memory for the image inside the processes own address space.
- Using the memcapy() function let the process copy its own image into the locally allocated memory.
- In order to allocate memory large enough to fit the image in the target process call VirtualAllocEx() function.
- Use the reloc table to calculate the offset for the image that was copied into the local memory.
- Modify all the absolute addresses to work at the address returned by VirtualAllocEx() by iterating the reloc table of the local image.
- By using WriteProcessMemory() function copy the local image into the memory region allocated in the target process.
- By subtracting the address of the function in the current process by the base address of the current process, and then adding it to the address of the allocated memory in the target process for calculating the remote address of the function to be executed in the remote process.
- Then CreateRemoteThread() create a new thread with the start address set to the remote address of the function.
- Now when the malicious PE is injected by the malware into another process, it will have a new base address which is unpredictable. So in that case the PE has to dynamically recompute the fixed address. To overcome this, the malicious code needs to find its relocation table address in the host process, and resolve

the absolute address of the copied image by looping through its relocation descriptors.

3 DEFENSE IN DEPTH

As the complexity of the malware increases, various security products cannot offer effective protection. Thus, a wide range of hardening procedure and security products are required to implement the robust defense system. Regular patch management of both operating system and third party application is important. Although this mitigate the effect of zero-day vulnerability but doesn't stop it. Continues monitoring the network, helps to identify the affected system which tries to receive the commands from the C2 server and exfiltrating the data.Strong access control mitigate the propagation of malware on the initial network. Use of Security information and event management (SIEM), intrusion detection system (IDS), intrusion prevention system (IPS), Firewalls plays an important role in the identification of the threats, setting up this devices would significantly increase the complexity of a successful attack. Control the use of USB's can also mitigate the propagation of the malware, limited use of such devices can is an important countermeasure. Use of good AV's also plays an important role in stopping the attack of malware over the host system.

4 RELATED WORK

In [1], the authors presented the high level overview of various malware and Security solutions failed to detect those threat. Malware authors often use the techniques like encryption and obfuscation to make their malicious code unidentifiable by the anti-virus software. By the use of this technique many security solution fails to detect the malicious program and that malicious program succeed to gain the persistence over the target machine. The author defines the complexity of identify the malware which were recently discovered running in the wild and has the ability to modify the system files and remotely connect to the attackers machine.

5 METHOD

In this research we will use the combination of static and dynamic analysis methods to identify the malware persistence technique. We will use the combination of few tool to identify the persistence mechanism [2] In the static analysis we analyze the malware without actually running it. Where as in dynamic analysis we analyze the malware sample by running it in the controlled and virtual environment and monitor the malware activity.

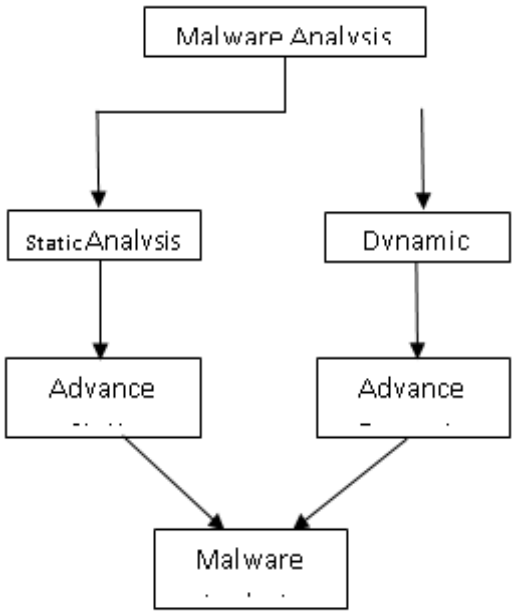

Fig. 5. *Methods for Malware Analysi*

PEid	Stands for executable identification. It is used to detect the PE is packed or obfuscated.
PEView	It is used to view the header and the various section of the PE.
Virtualbox	It is a virtually controlled platform. It can load multiple guest OS Under single host OS.
Wireshark	Wireshark is the packet monitoring tool, which helps to monitor outbound and inbound packets.
Ida Pro	IDA is the disassembler, help to disassembling the PE.
Dependency walker	It's a tool that scan the windows module (pe,dll,sys etc) and make the hierarchical tree diagram of all the dependent modules.
Olly debugger	Tool used for debugging and reversing the PE.
Regshot	Tool used to identify the various registry modified and added by the PE.

The safest method is consider as static analysis, as we don't execute the malware sample. The static analysis the sub dived into two basic static analysis and advance static analysis. Like that the dynamic analysis is also sub divided into two basic dynamic and advance dynamic analysis method. Tools and the malware samples used in the research process are listed below.

6 RESULT

For our research we have used several programs to analyze the malware. We have done experiments by running the malware sample and getting the output of the system. We have experimented with the malware samples by running them with few tools to identify their behavior.

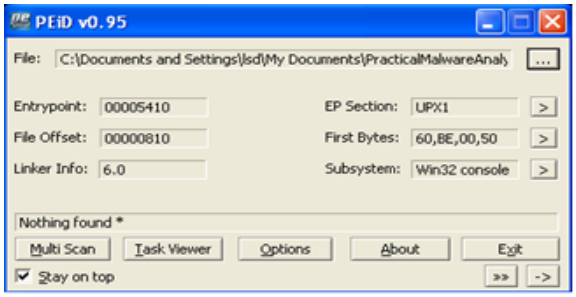

Fig. 6. *Tools & malware used in this research work.*

Fig. 7. *PeiD detector to identify the packers and the compiler of the PE file.*

The PEiD helps us to find the obfuscation technique used by the malware, once known we can deobfuscate it. PEiD also help us to find the compiler used to compile it. We can also see the entry point of the progr

Malware Sample Name	Hash value
LSD.exe	e2bf42217a67e46433da8b6f4507219e
RSA.exe	a7f21e412022554d187d6a876a3c08ac
pbkdf.exe	e2bf42217a67e46433da8b6f4507219e
ABC.exe	56bed8249e7c2982a90e54e1e55391a2

Fig. 9. *Analyzing the network traffic after running the LSD.exe*

As we started the packet capture and executed the malware sample it shows the network activity (all the inbound and outbound traffic) we get to know that malware LSD.exe tries to connect to a remoter server.

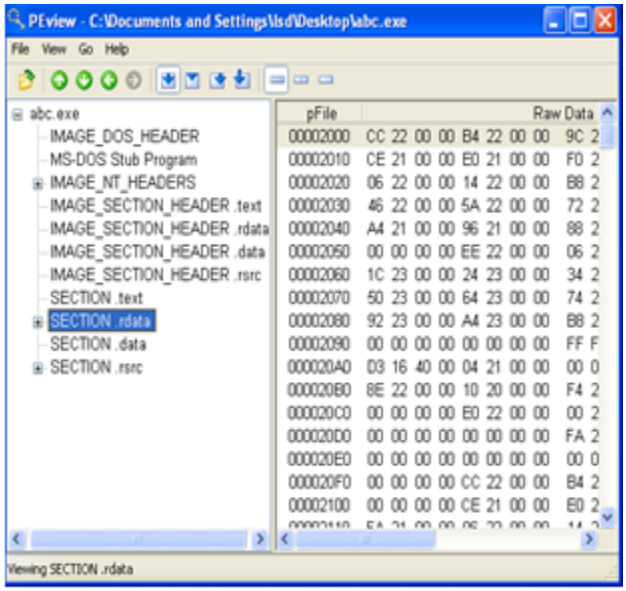

Fig. 8. *Analyzing the PE LSD.exe in the PEview*

In the PEview as we start the analysis, we can find the creation time of the malware however it can also be false.

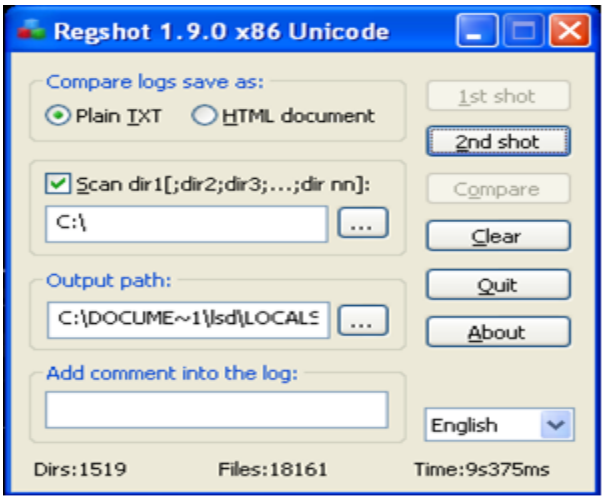

Fig. 10*. Regshot for the registry analysis of ABC.exe*

After running the ABC.exe we find that the malware modifies the various registry and also add few new registry in the system.

Fig. 11. *Disassembling the ABC.exe in IDA Pro*

When disassembling the ABC.exe in the IDA Pro we manage to find the functions like CreateMutexA() and OpenMutexA() which defines that the malware is using the mutex object for persistence.

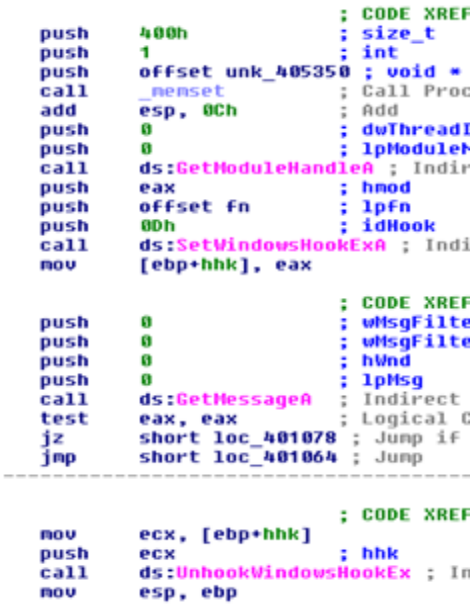

Fig. 12. *Disassembling the malware RSA.exe*

When we disassembled the malware RSA.exe we manage to identify the process hooking mechanism used by the malware. The function which helped us to find this were SetWindowsHookExA(), So in hear the malware tries to hook itself into a legitimate process.

Fig. 13. *Disassembling the pbkdf.exe in IDA Pro*

When we disassemble the malware pbkfd.exe in the IDA Pro find the function the functions VirtualAllocEX(), WriteProcessMemory() which are used for process injection. So we concluded that the malware pbkdf.exe uses the process injection technique for gaining the persistence over the host system.

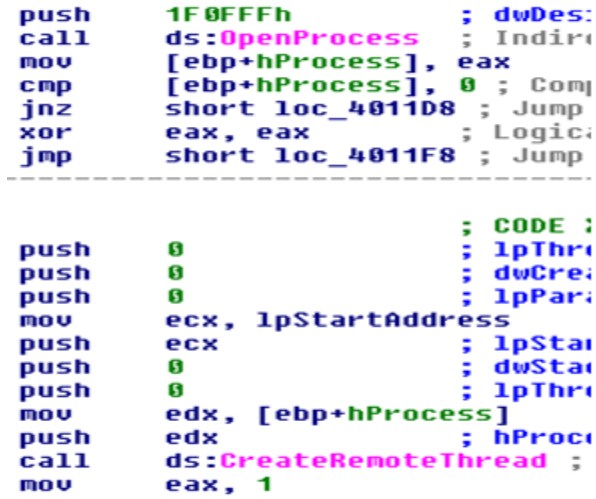

Fig. 14. *Disassembling the malware LSD.exe in the IDA Pro.*

When we started the disassembling of the malware deffie.exe we identified the a function named CreateRemoteThread(). So we concluded that the malware deffie.exe injecting its malicious code into the legitimate process to gain persistence.

Fig. 15. *Reversing the LSD.exe in OllyDbg*

While reversing the malware LSD.exe we find that the malware LSD.exe first tries to gain the privileges after gaining them is downloads a file from the remote server.
So we concluded that the winlogon.exe is the process injected. And sfc_os.dll is used to disable the windows file protection and acting as the persistence mechanism.

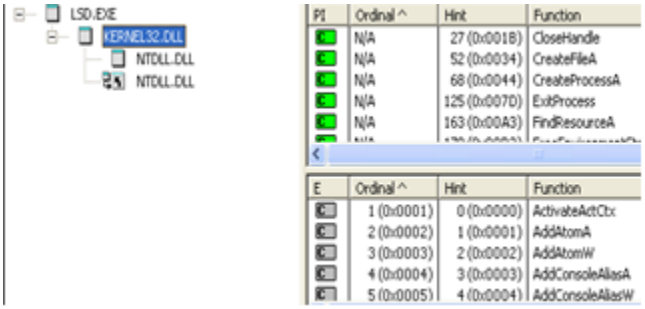

Fig. 16. *Identifying the DLLs loaded by LSD.exe*

Fig 16: LSD.exe in dependency walker.
When we used the dependency walker to analyze the imported dlls, we find that the malware LSD.exe imported the KERNEL32.dll and NTDLL.dll .

7 CONCLUSION

Throughout this research we have used the Combination of Advance static and advance dynamic methods of malware analysis to identify the various persistence mechanism used by the malwares to get the better results than the ordinary analysis method. In the research we have targeted to identify the various functions and files modified by the malware.Meanwhile, in the advance static analysis we manage to identify the various functions in the code, those function helps us to identify the various persistence mechanism used by the malware.On the other hand in the advance dynamic analysis we manage to identify the various files and registry modified which can be used as a persistence by the malware. We identified the modified registry, DLL imported, In the whole research we focused to identify the persistence mechanism used by the malware, we targeted to identify the functions while disassembling the malicious PE in the static analysis and advance static analysis and to identify the modified and newly created file in the dynamic and advance dynamic analysis process.And we concluded that the combination of static, dynamic and advance static and advance dynamic analysis process is the most suitable for identifying the malware persistence over the system.

8 REFERENCES

[1] Virvilis, Nikos, & Gritzalis, D**i**mitris. 2013. The Big Four- What we did wrong in Advance Persistent Threat detection? International Conference on Availability, Reliability and Security.
[2] Sikorski, Michael. Honig, Andrew. 2012. Practical Malware Analysis The Hands-On Guide to Dissecting Malicious Software. Publisher: No Starch Press.
[3] Distler, Dennis. 2007. Malware Analysis: An Introduction Technical Report: SANS Institute. December 2007
[4] Williams, Jake. DLL Hijacking Like a Boss!. Technical Report: SANS Institute
[5] Hosseini, Ashkan. 2017. Ten Process Injection Techniques: A Technical Survey Of Common And Trending Process Injection Techniques. Technical Report: Endgame. July 2017
[6] 0x1a0ran. 2013. Process Injection Monitor. January 2013
[7] Mansour, Yaser. 2014 An Early Malware Detection, Correlation, and Incident Response System with Case Studies Technical Report: SANS Institute. January 2014
[8] Brew, Kate. 2016 Malware Hiding Techniques to Watch for. Technical Report: AlienVault Labs. October 2016
[9] Zeltser, Lenny. 2015 “How Malware Generates Mutex Names to Evade Detection. Technical Report: SANS ISC InfoSec Forums. March 2015
[10] Stevens, Didier. 2010. DLL hijacking vulnerabilities Technical Report: SANS ISC InfoSec Forums. August 2010.
[11] www.attack.mitre.org/techniques/T1067/