Obfuscating Live Computer Forensic Investigative Process on a Windows 7 Operating System: A Criminal's Perspective

Dinesh Mothi
British Computer Society, Affiliate Member
Independent Digital Forensic Researcher
Secunderabad, India

ABSTRACT
Live forensic investigation is conducted when the computer system is turned on whilst the data is gathered in a forensically sound manner, from the physical memory, in the form of evidence. As time progressed, criminals have been developing methodologies by which live analysis could be defeated. One such method implemented by the criminals is that of a rookit being installed on the victim's machine. A rookit can be dangerous, and very risky to deal with from an investigator's point of view, because it has the power to subvert the kernel of an operating system. This paper presents, how easy it is for a criminal to thwart the process of live forensic investigation by downloading and installing free software tools; needing, no prior knowledge of the windows 7 operating system's kernel, and how frustrating it would be for the investigator to examine the computer system and make a valid forensic report. Thus, making live analysis a daunting task for the forensic investigator on field. Finally, a mathematical formula is derived for detecting the presence of hidden processes in the memory.

General Terms
Live Forensic Analysis, Physical Memory, Rootkit, Windows 7 OS.

Keywords
Anti-Forensics, Digital Forensics, Computer Forensics, Memory Forensics, Live Response.

1. INTRODUCTION
The windows operating system [20] is widely used than any other Operating Systems in the world, reason being, the ease and reduction in complexity by which a user can interact with the system. But, with its advantages comes the drawbacks. Due to its nature of operation, criminals take advantage of it and misuse the windows machine. It is not surprising to see new viruses or malware programs being created by the hackers to exploit the windows operating system. Hackers or Criminals develop malicious codes so as to hide their tools by which they carry out an attack on a victims system. Rootkits are toolsets used by an attacker to retain root-level access to a system in a covert manner [5]. The programme which hides the hacker tools that cause harm to the OS are known as rootkits, and with the widespread use of rootkits, attackers can conceal their activities [9]. In today's era rootkits are famous among the hacker community to thwart the process of live response methodologies on a windows machine. The forensic evidence is fragile to handle in the sense that it can be easily be modified, duplicated, restored or destroyed [8]. The Live Response Methodology, came into force when it was discovered that the traditional forensic methodology i.e., pulling the plug off the system and then to image the hard drive is no longer a viable option to acquire evidence from a machine. It was noted that the hackers used sophisticated software tools that would be installed in the volatile memory of a system and this would not write any of its contents to the hard drive. The idea behind this technique of running hacking tools from the volatile memory is that the data in the volatile memory vanishes when the power is taken away from the system. The volatile memory does not have the capability to retain or store any of its data once the power is turned off from it. With this in mind, the forensic investigators adopt the Live Response methodologies in order to collect data from the volatile memory such as Random Access Memory (RAM) from a system. The analysis of volatile memory data becomes an important aspect in live incident response, and there are a number of response toolkits being developed to address the needs [14, 16].

Volatile data is information that would be irrevocably lost if the machine suddenly lost power (e.g. the list of running processes, network connections, Logon sessions, etc.). Non-volatile data is persistent which is to say that we could acquire it from a forensic duplication of the machine's hard drive. The difference is that the format in which the information is conveyed is easier to read when requested from a running machine. With regard to collecting evidence, the prototypical forensic investigation normally proceeds according to the basic order of volatility spelled out by RFC3227 [19]. This sort of investigation begins with a live response process, where both volatile and non-volatile data are gathered [15].

The live response strategies that have been implemented so far are divided into, the data collection phase and data analysis phase. In the data collection phase, various information from the physical memory i.e., Random Access Memory (RAM) is collected -- process, network connections, clipboard contents etc. For forensic analysis, the collection of volatile information is such as Hardware information, Installed software packages or Process state is very important [12]. In data analysis phase, the data collected is analysed to determine if there is any unfamiliar activity taking place in the process or if any unwanted network ports are opened. The data analysis is done so as to know what kind of attack the computer system has been subjected to.

There are two types of live response methodologies on a windows system namely, local and remote. In a local live response methodology a forensic investigator has direct access to the computer system wherein by he can type commands in the console via keyboard and acquire data on a

Compact Disk (CD), thumb drive, or to any removable media which is local to the system. On the other hand, remote live response methodology commands are serially executed on a system across a network. This methodology is adopted when there are many systems to investigate because the process of logging into a system and running commands can easily be done without much effort.

The inconsistencies present in the memory violate the principles of digital forensics because data in the memory are not consistently maintained during system operation. This issue poses challenge for computer forensic investigator and need to be addressed before presenting the evidence in the court of law. [1, 2, 11]. The hackers and criminals nowadays constantly keep themselves abreast of the latest developments in the field of computer forensics. Blackhats have designed a procedure wherein they can easily defeat the live response methodologies. One of the strategies implemented by blackhats is to hide their presence when attacking a computer system. That is, the hacking tools are made to conceal themselves when they are installed on a machine. This strategy is implemented keeping in mind that the forensic investigator scans the physical memory of a machine in order to look for the presence of hacking tools.

2. LIVE ANALYSIS

Live analysis is one of the most important forensic investigative methodologies, adopted by forensic investigators, in order to acquire evidence from a computer system when it is powered on. Unlike the traditional forensic methodology, wherein the investigator pulls off the plug to turn the power off from the computer system to acquire evidence on the hard drive; the live analysis procedures are carried out on the random access memory i.e., the volatile memory of the computer system.

Digital forensics is divided into live analysis and dead analysis. The traditional forensic methodology focuses on not altering the time stamps on the computer system. In order to make sure that the time stamps are not modified, the forensic investigator photographs the compromised computer systems and its contents on the screen, and makes notes of hardware connections and their actions in their contemporaneous notes. Now, when the power plug is pulled off the system, the time stamps are preserved and are not changed. But, if observed closely whenever the power is shutdown the data in the volatile memory ceases to exist, and this is a vulnerability which exists in the traditional forensic methodology. Criminals or Hackers, have taken advantage of this loophole and have designed their tools accordingly to attack their victim's machine. The hacking tools are designed in such a way that they run in the physical memory (Random Access Memory) of the system which is volatile in nature. These specialised hacking tools do not write or get stored on non-volatile storage media of the system such as a hard drive. Therefore, with the advent of the hacking tools the method of pulling the plug of the machine to acquire evidence from it is not a viable option because the data in the physical memory is irrecoverably lost. The only difference between live and dead analysis is the reliability of the results. The same types of data can be analysed using dead and live analysis techniques, but the live analysis techniques rely on applications that could have been modified to produce false data [3].

The scenarios [18] where live analysis can be performed can be that of a computer system that is a victim of an intrusion. Since hacker tools frequently run only in system memory and leave no trace on the hard disks, the investigator now has to consider the fact that pulling the power plug may actually lose more evidence than it preserves. In such a case, touching the keyboard in order to extract and preserve evidence in the memory may be worth the cost of altering some system time. Memory Forensics is used by incident response handlers for the purpose of malware detection [7].

Live analysis involves extracting evidence from random access memory. The important steps to any live-analysis are as follows:

- Personally make the trusted tools and then bring them to the field.
- The interaction should be kept to a bare minimum on the compromised system.
- The investigator should be cautious whilst taking any actions on the live machine because the final events are unchangeable.
- Hashing all evidence and
- Gathering data in order of volatility [13].

If the machine is still active when arrived at the crime scene, we should collect the volatile information of victim of system rapidly, for example, which TCP and UDP ports are opened, user login history, what services are activated currently, etc. [4].

Conducting a live forensic examination involves a more complex approach than the traditional post-mortem examination. Care must be taken by the examiner on a live system to minimize the impact of any tools used. However, there is no way to avoid making changes, since in order to conduct a live examination it is necessary to deploy tools on the live system to capture data, and such tools will make changes to the running system [6].

Documentation is one of the most important steps in the live computer investigative process because, if the system is left on unattended, there will be changes made rapidly by the operating system which will have an impact on the physical memory. On the other hand, if the investigator begins his investigation, and is in the midst of collecting evidence from the physical memory, changes are still made to the volatile memory in which the evidence is located. This is the reason why the investigator has to maintain proper, accurate, and detailed notes at all times during the live forensic investigation[17].

3. RESEARCH WORK

3.1 Tools Used

- Operating System: Windows 7 Home Edition (32-bit).
- Kingston USB (Universal Serial Bus) 4 Gigabytes (GB).
- Advent Laptop (4GB RAM).
- ASMonitor.exe (keylogger).

3.2 Live Analysis on a Windows 7 Machine

To conduct a live analysis on a Windows 7 machine we need Live Analysis tools such as Windows Live Analysis CD's or Windows Live Analysis USB Drive. These tools consist of necessary software that is needed to recover evidence from the physical memory from the windows machine. The tools

contain dynamic link libraries, trusted tools, and open source tools in them.

3.2.1 Creating a Windows 7 Live Analysis USB Drive

1. Firstly, we need a brand new USB Drive. A 4GB Kingston USB Drive is used in this research work.

2. Now a genuine Windows 7 Operating System should be installed on a clean Hard Drive. For this purpose a brand new Laptop was used on which the Windows 7 Operating System was installed.

3. Open the C Drive and navigate to the system 32 folder. For this the following steps have to be followed:

- Open or Click on the My Computer icon. Then we will come across a few Drives or partitions.
- The primary partition or the Drive on which the Windows 7 is installed, by default it will be the C Drive.
- Open the C Drive by clicking on it and then a number of folders will be found on it.
- Click on the Windows Folder. Then, when the Windows folder opens, click on the system 32 Folder.
- Now when the system 32 Folder is opened. The .dll files should be copied from it onto the USB Drive. We should make sure that we login as administrator to perform this task. (The laptop is set to login as administrator by default).
- Now, on the right hand side, top of the screen we will find a search option. Type in *.dll in the search box. This will show us all the .dll files present in the system 32 folder which is needed to be copied onto the USB Drive.
- In order to copy the .dll files. Press CTRL+A, this selects all of the .dll files. Then press CTRL+C this copies all the selected .dll files. To paste these files onto the USB Drive, just open the USB Drive and press CTRL+V.
- After the .dll files have been copied onto the USB drive, the next step is to copy a few .exe files on to the USB Drive. The .exe files such as WHOAMI.exe, TASKLIST.exe, etc., should be copied onto USB Drive because these tools will help an investigator to find evidence in the physical memory. For instance, the WHOAMI tool gives us the name of the computer system and the user who is logged on into it.

The TASKLIST can give you the number of processes residing in the memory along with the Process Identifier (PID) and how much memory that each process consume. Also, open source tools which are freely available on the internet can be used for the live analysis purposes. We can add these tools on the Live Analysis USB Drive. The tools that are downloaded from the internet must first be thoroughly tested under real time conditions before taking them on the trusted tool thumb drive for investigation purposes.

3.3 Defeating Live Forensic Investigation on the Windows7 OS

The Software HideWizard.exe is used for this purpose. Although, this free software tool's main purpose is not intended to defeat the live forensic investigation, but by taking advantage of its features, it can be used to obfuscate the live investigation being carried out on the field by the investigator. This software tool can hide files, folders or even processes that can go undetected if traditional live analysis methods are followed. Now, let us use the Live Analysis USB Drive to conduct a Live Analysis on the Windows 7 machine. The steps are as follows:

- First insert the USB drive into USB slot of the computer.
- Then, open My Computer. The USB drive is usually the F Drive.
- Click on the F drive to open the USB Drive (Live Analysis USB Drive.).
- Click on the #cmd.exe to open it (Trusted Tool Command Prompt.).

The Live Analysis USB Drive consists of various .dll files and other .exe files which will play a vital role when conducting the live computer forensic investigation. Since, these files are present on the computer forensic investigator's USB drive; they become a part of the trusted tool set. This means the investigator will have a safe and a secured provision to commands from his trusted USB drive rather than running those same commands from the compromised system. This has a few advantages because the evidence in the random access memory will not be altered, whereas if the same commands are run from the compromised computer system, the evidence present in the physical memory will be changed. Also, it would be a good practice if the investigator has a trusted USB device with storage capacities twice of that of the physical memory. The reason behind this is, that the investigator has to copy back the evidence located in the memory onto his trusted USB drive. In the trusted tool command prompt type in "tasklist" to view the processes that are residing in the physical memory. By using the HideWizard software tool we can hide any number of the processes that is in the physical memory. For example, say if we want to hide chrome.exe which is a Google chrome web browser application:

- Open HideWizard.
- Click on the settings button (Hide Wizard Settings)
- On the left, we can see a "File and Process" button, which is present on the bottom left of the software tool. Click on it.

Now, in order to hide a process, just enter the name of the process it in the "hide these processes while Hide Wizard is running" box. For instance, to hide Google chrome process, enter "chrome.exe" in the box to hide.

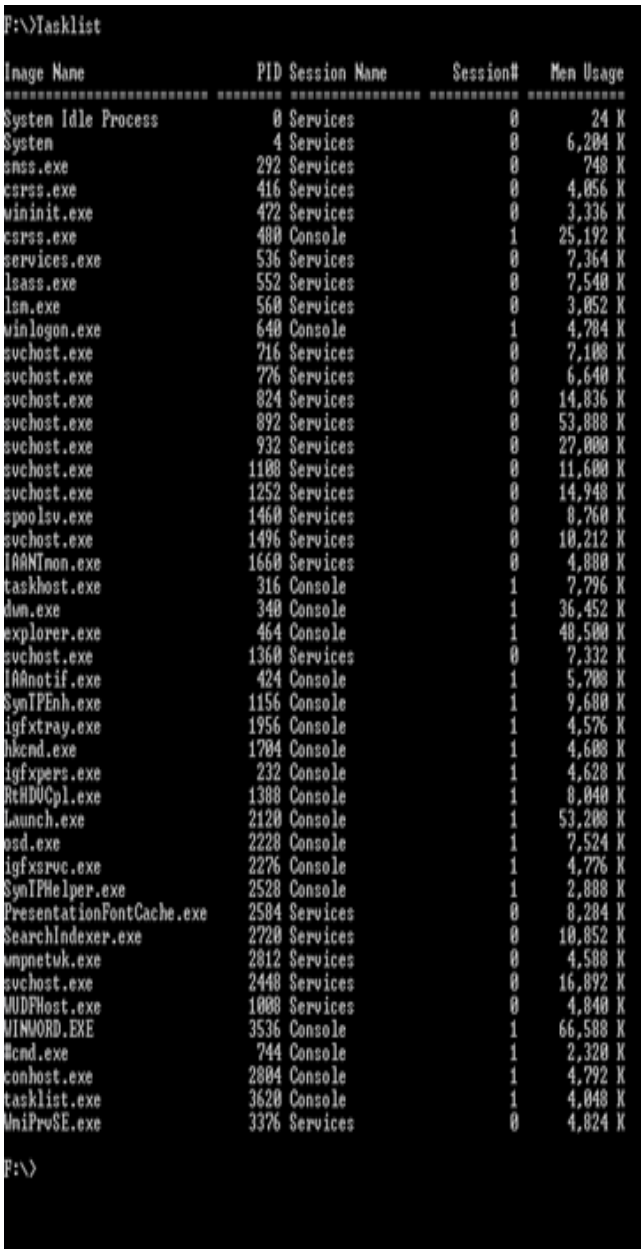
Now, type in the "tasklist" command in the command prompt. We will not be able to find chrome.exe process in it.

Let us hide another process ASMonitor.exe which is a keylogger in the physical memory. A keylogger is a potential hacking tool. Follow the above procedure to hide it. Instead of chrome.exe just type ASMonitor.exe in the box in order to hide the process. Now, again type the "tasklist" command in the command prompt and the process ASMonitor.exe is hidden.

This Software tool has the ability to hide itself. Just type HideWizard.exe in the box and then when the command "tasklist" is typed in the command prompt, the process HideWizard.exe is hidden in the memory (Figure 1).

Live Analysis has been defeated with the help of free Software Tool HackWizard.exe which acts like a rootkit. It can hide itself and also can other processes in the memory. This proves that an attacker need not comprehend the internal kernel architecture of the windows 7 operating system in order to design a rootkit from the scratch which would prove to be quite tedious in implementation when compared to using

a free graphical user interface tool which is user friendly and would be easy to use for a novice attacker who does not understand the intricacies of windows 7 operating system.



```
F:\>Tasklist

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0         24 K
System                           4 Services                   0      6,204 K
smss.exe                       292 Services                   0        748 K
csrss.exe                      416 Services                   0      4,056 K
wininit.exe                    472 Services                   0      3,336 K
csrss.exe                      480 Console                    1     25,192 K
services.exe                   536 Services                   0      7,364 K
lsass.exe                      552 Services                   0      7,540 K
lsm.exe                        560 Services                   0      3,052 K
winlogon.exe                   640 Console                    1      4,784 K
svchost.exe                    716 Services                   0      7,108 K
svchost.exe                    776 Services                   0      6,640 K
svchost.exe                    824 Services                   0     14,836 K
svchost.exe                    892 Services                   0     53,888 K
svchost.exe                    932 Services                   0     27,800 K
svchost.exe                   1108 Services                   0     11,600 K
svchost.exe                   1252 Services                   0     14,948 K
spoolsv.exe                   1460 Services                   0      8,760 K
svchost.exe                   1496 Services                   0     10,212 K
IAANTmon.exe                  1660 Services                   0      4,880 K
taskhost.exe                   316 Console                    1      7,796 K
dwm.exe                        340 Console                    1     36,452 K
explorer.exe                   464 Console                    1     48,500 K
svchost.exe                   1360 Services                   0      7,332 K
IAAnotif.exe                   424 Console                    1      5,708 K
SynTPEnh.exe                  1156 Console                    1      9,680 K
igfxtray.exe                  1956 Console                    1      4,576 K
hkcmd.exe                     1704 Console                    1      4,608 K
igfxpers.exe                   232 Console                    1      4,628 K
RtHDVCpl.exe                  1388 Console                    1      8,040 K
Launch.exe                    2120 Console                    1     53,208 K
osd.exe                       2228 Console                    1      7,524 K
igfxsrvc.exe                  2276 Console                    1      4,776 K
SynTPHelper.exe               2528 Console                    1      2,888 K
PresentationFontCache.exe     2584 Services                   0      8,284 K
SearchIndexer.exe             2720 Services                   0     10,852 K
wmpnetwk.exe                  2812 Services                   0      4,588 K
svchost.exe                   2448 Services                   0     16,892 K
WUDFHost.exe                  1008 Services                   0      4,840 K
WINWORD.EXE                   3536 Console                    1     66,588 K
Wcnd.exe                       744 Console                    1      2,320 K
conhost.exe                   2804 Console                    1      4,792 K
tasklist.exe                  3620 Console                    1      4,048 K
UniPrvSE.exe                  3376 Services                   0      4,824 K

F:\>
```

Figure 1: HideWizard.exe hidden

4. DETECTION OF HIDDEN PROCESS

The traditional method to detect hidden processes in the memory is carried out by examining the physical memory dumps, but in this research work, the detection of hidden process is shown by the Mathematical Analysis of various processes running in the Random Access Memory.

Methodology: Derivation of the Mathematical Formula to find out if a process is hidden in the physical memory: Windows 7 OS has an inbuilt tool Resource Monitor, Resmon.exe (Figure 2) that can show the processes active in the physical memory along with the Process Identifier (PID), and with different types of memories which is associated with the process.

To open the Resmon tool go to the start menu and type Resmon.exe. When Resmon tool is opened we will find all the process with their corresponding memory usage. The process memory is divided into the commit and the working set memory.

Commit charge is the total amount of pageable virtual address space for which no backing store is assigned other than the pagefile.

The working set of a program is a collection of those pages in its virtual address space that have been recently referenced. It includes both shared and private data. The shared data includes pages that contain all instructions your application executes, including those in your Dynamic Link Libraries (DLLs) and the system DLLs. As the working set size increases, memory demand increases. The Working Set is further divided into Sharable Working Set and Private Working Set.

Private Working Set: Resident pages which are private only to this process.

Shared Working Set: Resident pages which are currently being shared with other processes. This is a subset of the Shareable Working Set.

Experiment 1: The factor by which the InUse Memory increases when a process is added in the physical memory. Here a process used by Google Chrome web browser chrome.exe is used for the experiment. The results are calculated before and after chrome.exe is run in the physical memory.

Before Chrome.exe:

The InUse Memory varied from 695 MB to 705 MB.

After Chrome.exe:

The InUse Memory varied from 750 MB to 764 MB.

When Google Chrome is run it adds two chrome.exe processes in the memory. The relation between InUse Memory, and Commit Charge and Working Set is:

The Difference between the Low Values of InUse Memory (i.e. the low values if InUse Memory before and after chrome.exe) = 750-695 = 55 Megabytes (MB).

The Difference between the High Values of InUse Memory = 764-705=59 Megabytes (MB).

Sum of the Values of Commit Charge and Working set for chrome.exe (1) in Megabytes (MB) is (30880/1024) + (35664/1024) =30.1 MB + 34.8 MB = 64.9 Megabytes (MB).

Sum of the Values of Commit Charge and Working Set for chrome.exe (2) in MB is (19524/1024) + (35024/1024)

=19.0 MB + 34.2 MB = 53.2 Megabytes (MB).

Mean of Values of Chrome.exe (1) and Chrome.exe (2) = (64.9 + 53.2)/2 = 59.05 Megabytes (MB).

This experiment was repeated with applications like Acrobat reader, command prompt etc., and the following observations were made:

- Whenever a new process is added to the physical memory, the physical memory increases by a factor which is equal to the mean of the values of the commit and working set of that process.
- The mean of commit charge and working set of a process varies between the difference of low and high values of InUse Memory, before and after, that particular process has been added.

- The Values of Commit charge and Working set for most of the processes are not constant, and keeps varying. Hence, bringing variations in InUse Memory and the processes in the physical memory.

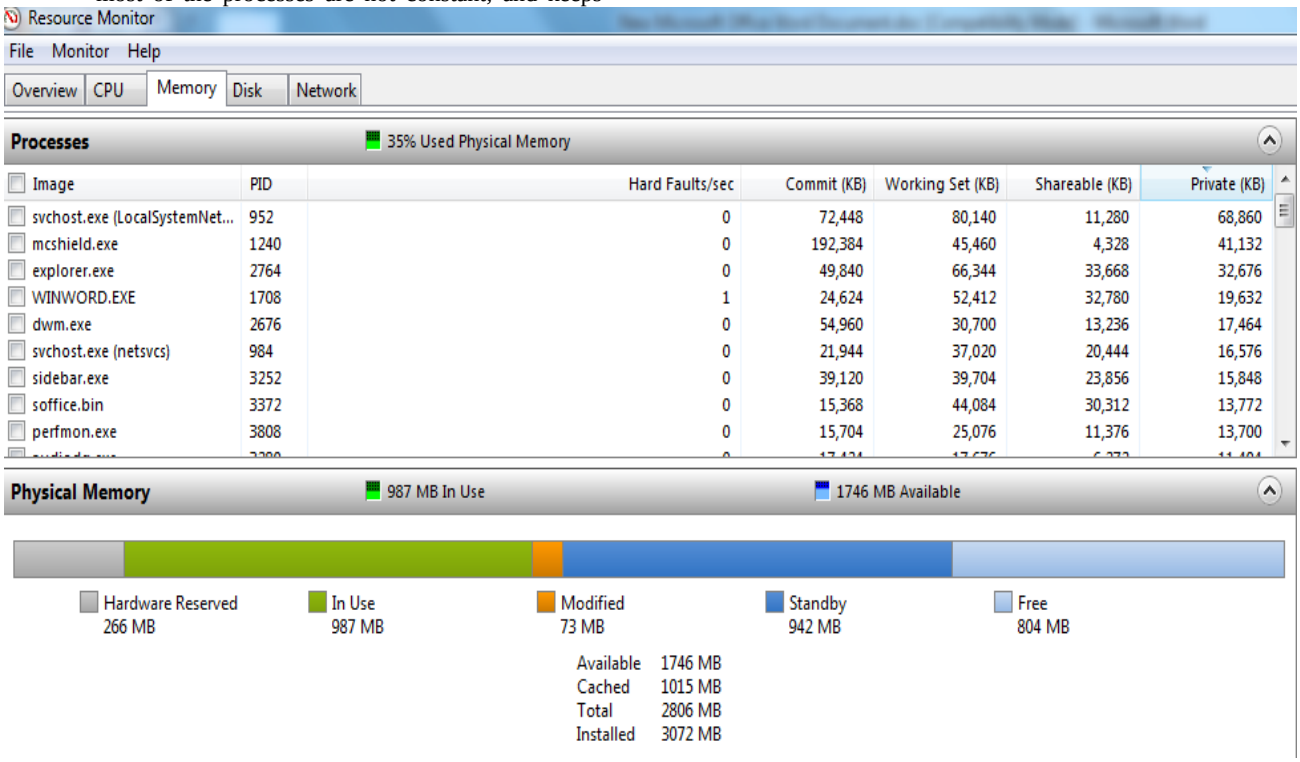


Figure 2: Resmon.exe

4.1 Derivation of the formula for InUse Memory

From the observations made from the experiment it is known that the processes that reside in memory are variable in nature. Therefore, mathematically the rate at which a process (P_r) changes is given by dP_r/dt. This is known as the rate of change of a process (P_r) with respect to time (t).

In order to find the value of a process (P_r) at a particular instant of time say 'T', let 'Ip' denote the instantaneous value of the process:

Then, $I_p = \int_{st1}^{t2} \Pr(t).dt$ ----------> Equation (1).

Therefore, the instantaneous value of a process is the definite integral of the process (p) which is a function of time (t). The time 'st_1' and 't_2' denotes the time at which the process starts and terminated. Also, $T = t_2\text{-}st_1$.

Let M_{Int} denote the InUse Memory at a particular instant of time. Therefore, if there are 'n' number of processes in the memory then,

$M_{Int} = A + I_{p1} + I_{p2} + I_{p3} + \ldots\ldots\ldots\ldots + I_{pn}$ ----------> Equation (2).

Where, A = constant.

I_{p1}, I_{p2}, I_{p3} = Instantaneous value of the 1st process, 2nd process, 3rd process and so on.

Substituting Equation (1) in Equation (2), we get:

$M_{Int} = A + \int_{str1}^{tr1m} \Pr1(t).dt + \int_{str2}^{tr2m} \Pr2(t).dt + \int_{str3}^{tr3m} \Pr3(t).dt + \cdots\ldots\ldots\ldots\ldots + \int_{strn}^{trnm} \Prn(t).dt$ ----------> Equation (4).

Where,

A = constant.

Pr1, Pr2, Pr3, ..., Prn = 1st Process, 2nd Process, 3rd Process, ..., nth Process.

(str1, tr1m) = start and termination time for process1.

(str2, tr2m) = start and termination time for process2.

. . . . . .

. . . . . .

(strn, trnm) = start and termination time for process n.

Therefore,

$M_{Int} = A + \sum_{i=1}^{n}[\int_{stri}^{trmi} Pri(t).dt]$ ----------> Equation 5.

From Experiment 1 we have for any given process added to the physical memory, the physical memory gets increased by a factor which is the mean of the commit charge(C) and working set (W) values, i.e. P = (C+W)/2.

Therefore, for

Process1, $Pr1 = (C_1+W_1)/2$

Process2, $Pr2 = (C_2+W_2)/2$

Process n, $Prn = (C_n+W_n)/2$

Now, substituting the values of Pr1, Pr2... Prn in Equation 5 we get,

$M_{Int} = A + \sum_{i=1}^{n}[\int_{stri}^{trmi}[(Ci + Wi)/2](t).dt]$

$M_{Int} = A + 1/2\sum_{i=1}^{n}[\int_{stri}^{trmi}[Ci + Wi](t).dt]$ ----------> Equation 6.

Where,

A= Constant.

Ci = Commit of the i^{th} process.

Wi = Working set of the i^{th} process.

(stri, trim) = start and termination time for the i^{th} process.

M_{Int} = Instantaneous value of the InUse Memory.

Equation 6 can be used to verify or check whether if processes (in Megabytes (MB)) are hidden in physical memory or not. For this, two things should be understood clearly:

- When a process is terminated or deleted.
- When a process is hidden in the physical memory.

When a process is terminated the Value of the InUse memory is reduced by a factor of (C+W)/2. Whereas, when a process is hidden it has no effect on the InUse Memory. So, when the resmon tool is opened the value of the InUse Memory does not change if a process is hidden in the memory. If a process is hidden only its name just disappears from the process list. But, it will not affect the value of the InUse Memory. Therefore, when the value of InUse Memory is calculated by using Equation 6 it should be equal to value of InUse Memory in the Resmon tool, then no process is hidden in the Physical memory.

Let M_{Inr} be the value of InUse Memory in the Resmon Tool. And, M_{Int} is the value of InUse Memory in Equation 6. Now two Cases exist:

Case 1:

$\mathbf{M_{Int} = M_{Inr}}$. If these two values are equal then no process is hidden in the physical memory.

Case 2:

$\mathbf{M_{Int} < M_{Inr}}$. If the value of M_{Int} is less than M_{Inr} then a process or processes are hidden in the physical memory.

Also if,

$\mathbf{M_{Inr} - M_{Int} = K}$. Where 'K' is a constant which shows the memory of the hidden process in Megabytes (MB). This means that a process of K MB of memory is hidden in the physical memory.

If $\mathbf{M_{Inr} - M_{Int} = 0}$, then there is no process hidden in the memory.

The mathematical analysis could form the basis for the development of the software tools in order to detect any hidden processes in the physical memory. A software developer can take the help of the mathematical formula in order to write algorithms, construct flowchart, write pseudo codes and then implement the functionality with the help of a programming language.

5. CONCLUSION

The live forensic investigation process is defeated on a windows7 computer system by downloading and installing Hide Wizard free software tool from the internet. These free software tools are primarily intended to hide files on a computer and not for the purpose of defeating live analysis. But, by taking advantage of what these tools got to offer, a criminal can easily defeat the live investigation process on a windows7 machine. This would be the most preferred method for criminals, especially those who are not tech savvy to defeat live analysis rather than designing a malicious software tool such as a rootkit; reason being that, development of a rootkit requires lot of time, money and tedious labour to come out with the final product, because this would involve various steps ranging from understating the windows Kernel thoroughly, choosing appropriate programming languages and testing the tool to verify its functionality. Whereas, the free software tools are user friendly due to the Graphical User Interface features. Therefore, the hacker can get accustomed and familiarize himself with these tools within a less period of time. The mathematical formula which was derived to find if a process is hidden in the Random Access Memory could form the basis to develop a hidden process detection software tool.

6. REFERENCES

[1] Thomas Sudkamp (1986), Inference propagation in emitter, system hierarchies, Proceedings of the ACM SIGART International Symposium on Methodologies for Intelligent Systems, ACM Press New York, NY, USA, pp 165-173

[2] Amihai Motro, Philipp Anokhin and Aybar C. Acar (2004), 'Utility-based Resolution of Data Inconsistencies Information Quality in Informational Systems', Proceedings of the 2004 international workshop on Information quality in information systems, ACM Press New York, NY, USA, pp 35-43.

[3] Brian D. Carrier 2006, 'Risks of Live Digital Forensic Analysis', Communications of the ACM, Vol. 49, No. 2, pp.56-61.

[4] Pei-Hua Yen, Chung-Huang Yang, Tae-Nam Ahn (2009), 'Design and Implementation of a Live-analysis Digital Forensic System', International Conference on Convergence and Hybrid Information Technology, Proc. ACM, pp. 239-243.

[5] John G. Levine, Julian B. Grizzard and Henry L. Owen (2006), 'Detecting and Categorizing Kernel-Level Rootkits to Aid Future Detection'. In IEEE Security and Privacy, Proc. ACM pp 24-25.

[6] Iain Sutherland , Jon Evans , Theodore Tryfonas , Andrew Blyth (2008), 'Acquiring volatile operating system data tools and techniques, ACM SIGOPS Operating Systems Review, v.42 n.3, pp. 65-73.

[7] Luka Milkovic (2012) 'Defeating Windows Memory Forensics', INFIGO Information Security, Available at: http://www.youtube.com/watch?v=RPVmLhP7K6U (Accessed 18th January 2013)

[8] S. Mocas (2003) 'Building Theoretical Underpinnings for Digital Forensics Research', Portland State University, Digital Investigations, pp. 1-10.

[9] D. Dittrich (2002) 'Root Kits and Hiding Files/Directories/ Processes after a Break-In', Available at: http://staff.washington.edu/dittrich/misc/faqs/rootkits.faq . (Accessed 5th August 2011)

[10] B. Carrier (2009) 'The Sleuth Kit and Autopsy', Available at: http://www.sleuthkit.org/autopsy/desc.php (Accessed 18th August 2011)

[11] B. Westbrook and B. Zornado (2001), Proposal for electronic records management task force, Available at: http://www.uclibraries.net/sopag/erm/ERMTFReport.pdf (Accessed 27th September 2011)

[12] C. Pogue, C. Altheide and T. Haverkos (2008), UNIX and Linux Forensic Analysis DVD Toolkit, Syngress Publishing.

[13] F. Adelstein (2006), Live forensics: diagnosing your system without killing it first, Communications of the ACM, Vol.49, No.2, pp. 63-66.

[14] Harlan Carvey (2005), 'Windows Forensics and Incident Recovery', Addison Wesley, Burlington.

[15] Bill Blunden (2009), The Rootkit Arsenal: Escape and Evasion, Jones and Bartlett Learning, Texas.

[16] Kevin Mandia, Chris Prosise, and Matt Pepe (2003), 'Incident Response and Computer Forensics', McGraw-Hill Osborne Media, 2 edition.

[17] Information Security and Forensics Society (ISFS) (2004), Computer Forensics, Part 2: Best Practices, Available at: http://www.isfs.org.hk/publications/ComputerForensics/

ComputerForensics_part2.pdf (Accessed 21th August 2011)

[18] Steve Anson and Steve Bunting. 2007 Mastering Windows Network Forensics and Investigation. Wiley Publishing Inc.

[19] Internet Engineering Task Force, url: https://www.ietf.org/rfc/rfc3227.txt

[20] Microsoft, url: http://www.microsoft.com/en-us/windows