ELSEVIER

Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 29 (2012) 1253 – 1258

Procedia Engineering

www.elsevier.com/locate/procedia

2012 International Workshop on Information and Electronics Engineering (IWIEE)

A Probability Model of Covering Key Trace during Ca[illegible]ring Volatile Memory

Lianhai Wang, Hengjian Li*, Zhen Su

Shandong Provincial Key Laboratory of Computer Network, Shandong Computer Scienc[illegible]ter, jinan, [illegible]014,C[illegible]

Abstract

In this paper, we give a clear description of the running memory ac[illegible]ing tool on a target system, especially for possibility covering the key trace during capturing volatile memor[illegible]ome key tr[illegible] of offender may still in the running memory after the scene of a crime and have critical role in [illegible]t and secu[illegible] applications. However, some key trace, such as rootkits in the memory, memory occupied by their co[illegible]onding [illegible]cess will probably be covered/ reallocated during the procedure of obtaining evidence of [illegible]rime. There[illegible]overed ratio (lost data) should be evaluated and investigated after the forensic tools run. F[illegible]y, [illegible]del the distribution of key trace exacted in the unallocated memory space, then form a formula to eva[illegible]te th[illegible]e rate of the key trace in which the corresponding process has just been killed. At last, we give [illegible]es to analyze the evidence coverage ratio which can be estimated by the new allocated memor[illegible]

Keywords: Computer forensics; k[illegible] memory cap[illegible]; Microsoft Windows XP SP2

1. Introduction

Physical [illegible]ory [illegible]ensics has gained a lot of attention over the past years[1], and various methods have been investi[illegible] for ca[illegible]ring and examining the volatile storage of a target machine. Memory analysi[illegible]ven [illegible]ffic[illegible] during incident response and more traditional forensic investigations. And, we c[illegible] extra[illegible]useful in[illegible]ation from the physical memory, such as extracting the content of windows cl[illegible]ard fr[illegible]ysical memory[2], command lines in the DOSKEY[3] and registry information [4,5]. Ther[illegible]wo exis[illegible] memory acquired methods, one is based on hardware, and the other is based on softwar[illegible] The hardware method needs installing the related device before, which is not useful in the

* Corresponding author. Tel.: +0-86-13515319380
E-mail address: lihengj@keylab.net

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2012.01.122

real cases. Therefore, the software-based acquired memory tools are actually employed in digital investigation. However, when running the software-based acquired memory tools, uploading the related driver files would probability alter the content of target memory.

On one hand, the volatility memory changes as the OS is running in a natural way. On the other hand, the memory content is varied by the uploading drivers when capturing the target machine memory. During a live forensics-investigation one rule of best practice is to minimize the impact on the target system, and it is an unavoidable action. And this leads to the trustable of acquired memory, that is, whether the evidence information is changed or covered because of running a memory acquired process. At present, the main research in memory forensics is how to extract information from memory as much as possible, such as network connection state, registry and login password. However, the effect of memory captured tool on the system has not well been investigated yet. In [6], Aaron Waters discuss the extent of memory changes over time. On the basis of a comprehensive summarizing and reviewing of existing results, Wang proposed a novel Model of Computer Live Forensics [7], and firstly took the credibility of digital evidence as a starting point, the issue of credibility of live forensic is then put forward for study.

Memory analysis is a key element of digital forensics. Leveraging memory to determine the state of the machine at the time of the incident is often critical to success. While there has been significant research into memory forensics, to date there has not been research into covering the key trace during capturing memory. The memory content of a process is not lost immediately when the process is killed. The acquired memory tool may cover or disturb this part of memory. In this paper, we estimate the cover possibility for the loading acquired tool.

The remainder of the paper is organized as follows. In Section 2 we briefly describe the memory management process and give an overview of the most important data structures that are required for this task. Modeling the Covering key trace of evidence is proposed in section 3. And the probability computation algorithm of covering on key trace memory zone is also described in detail in this section. Experimental results and analysis is presented in section 4. We conclude with a summary of our work and indicate opportunities for future research in this area in section 5.

2. Process and virtual address to physical address in memory

Based on Kernel Processor Control Region (KPCR), Zhang propose an effective process analysis method which uses a combination of scanning and list traversing techniques[8]. Brendan Dolan-Gavitt proposed that the Virtual Address Descriptor (VAD) tree, can provide such an abstraction layer over the page directory and page tables by describing the memory ranges allocated by a process as they might be seen by the process – as mapped files, loaded DLLs, or privately allocated regions[9]. VADs are data structures linked to each process's EPROCESS block that the memory manager uses to keep track of which virtual addresses have been reserved for each process. Programs usually operate on virtual memory regions only, therefore, to manipulate the respective physical data, the Memory Manager must continuously translate (map) virtual into physical addresses[10].

This virtual to physical addresses procedure works as follows: At the hardware level, volatile storage is organized into units called pages. A common size of such pages is 4kB on x86-platform. To reference them, the operating system implements a two-level approach: For every process, the operating system maintains a page directory that saves pointers (Page Directory Entries PDEs 4 bytes each, containing a pointer and several flags) to 1024 links (Page Table Entries, 4 bytes each) to the corresponding pages in the main memory. Thus, in order to translate a virtual to a physical address, the memory manager first needs to recover the base address of the page directory. It is stored in the CR3 registry of the processor and is reloaded from the _KPROCESS block of the process at every context switch. The first 10 bits of the virtual address can then be used as an index into the page directory to retrieve the desired PDE. With the help of the PDE and the page table index, i.e, the subsequent 10 bits of the virtual address, the page table and PTE in question are identified in the next step. To find the

RETRACTED

approximate page and data in RAM eventually, the PTE and the 12-bit byte index of the virtual address are parsed.

The Virtual Address Descriptor tree is used by the Windows memory manager to describe memory ranges used by a process as they are allocated. When a process allocates memory with VirutalAlloc, the memory manager creates an entry in the VAD tree. A node in the tree is associated with a pool tag and is of type _MMVD_SHORT("VadS"), _MMVAD("Vad"), or _MMVAD_LONG("VadL"). The latter two store a pointer to a _Control_Area structure that, in turn, points to a _File_Object that holds the unique file name. Consequently, the entire list of loaded modules can be retrieved by traversing the VAD tree from top to bottom and following the corresponding _Control_Area and _File_Object references.

3. Model and Evaluation the Covering key trace of evidence

3.1. Model and Evaluation the influence of the key trace caused by acquiring memory to[illegible]

When performing memory analysis, there are two primary components: a) kernel memory and b)userland memory. In a Windows environment, kernel memory is comprised primarily of device drivers, the NT operating system executable, and HAL.DLL. The majority of userland memory is composed of individual process, and the process' address primarily consists of files loaded from disk that contain code or data needed for process execution, typically Portable Executables (EXEs) and Dynamic Link Libraries(DLLs). Most of the virtual address in kernel memory is global regardless of the process context, whereas the other virtual address space is generally specific to each particular process.

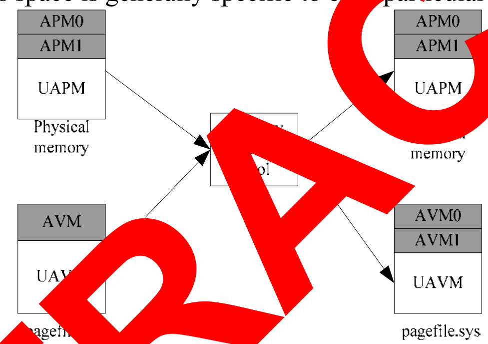

Fig.1 Effect on the target system memory while running acquiring tool

Techniques used in live forensics will inevitably change the system under investigation because they must be conducted by running tools on the system. And as such any findings may be problematic as evidence at court. Effects of live forensics to collected evidence included the probability of covering key trace by forensic tool kit and affecting region in digital evidence. As showed in Fig.1, with the running acquiring tools, some of the unallocated memory will be allocated for the new process. And, some memory data may be transferred into the pagefile.sys.

Walking the VAD tree is simply a matter of identifying the _EPROCESS structure for the process of interest, locating the VadRoot member (0x11c in all versions of XP), and then following each link to the left and right [illegible] until the entire tree is traversed. All addresses are virtual, so the page directory for the process is also needed in order to successfully read the tree. For affecting region, it can be obtained by finding physical memory occupied by forensic tool through its VadRoot. The probability of covering key trace can be calculated by probability statistics, which can be inferred in the following section as follows.

(1) Supposed the key traces distributed uniformly in the unallocated memory pages. The number of the unallocated memory page is m, and the symbol of these pages noted as $\Omega_m = \{B_1, B_2, \cdots, B_m\}$ 。

(2) The pages of the unallocated memory increased as running the forensic tools. After the memory acquired tool, supposed the number of novel unallocated page is n (of course, n<m). We noted these pages as $\Omega_n = \{B'_1, B'_2, \cdots, B'_n\}$ ($\Omega_n \subset \Omega_m$). Then, we can simply treat the $\Omega_r = \Omega_m - \Omega_n$ as the forensics tools occupied pages. And we noted the number of the containing the key traces memory pages as ***e*****,** of course $\Omega_e \subseteq \Omega_r$.

(3) Suppose the event A'_x = "the key trace contained in Ω_r". Therefore, it can be inferred as $P(A'_x) = \dfrac{e}{m-n}$ .This formulation approximate the key trace probability of the arbitrarily memory page contained in Ω_n. That is,

$$P(A_x) = P(A'_x) = \frac{e}{e+m-n}$$

(4) The probability of covering the key traces in unallocated memory can be computed as:

$$p(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i) - \sum_{1\le i_1 < i_2 \le n} P(A_{i_1}A_{i_2}) + \cdots + (-1)^{k-1} \sum_{1\le i_1 < i_2 < i_k \le n} P(A_{i_1}A_{i_2}\cdots A_{i_k}) + \cdots + (-1)^{n-1} P(A_1 \cdots A_n) \quad (2)$$

$P(A_x) = \dfrac{e}{m-n}$ is replaced with above formula and it can be simplified as :

$$P(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} (-1)^{i-1} C_n^i (\frac{e}{m-n})^i = 1-(1-\frac{e}{m-n})^n \quad (3)$$

3.2. The flow char of the computing the key trace covering probability

Virtual address	physical address	size,	Virtual address	physical address	size
0x370000	0xf108000	0x1[illegible]	[illegible]1000	0x92e6000	0x1000
0x371000	0xf109000	[illegible]00	[illegible]2000	0x9367000	0x1000
0x372000	0xf0a7000	[illegible]000	0x[illegible]3000	0x1e3a8000	0x1000
0x373000	0xf0a[illegible]	0x[illegible]	[illegible]3f0000	0x18ebd000	0x1000
0x400000	0x[illegible]000	0x100[illegible]	0x3f1000	0x92be000	0x1000
0x401000	[illegible]	0x1000	0x400000	0x17b12000	0x1000
0x402000	0xf174[illegible]	0x1000	0x401000	0xa4a2000	0x1000
0x40[illegible]	0xf175000	[illegible]000	0x402000	0x11663000	0x1000
[illegible]000	0xf176000	[illegible]00	0x403000	0x13de4000	0x1000
[illegible]405000	0xf11b000	0x1000	0x404000	0x548c000	0x1000
[illegible]	0xf0e[illegible]	0x1000	0x405000	0x648d000	0x1000
0x4[illegible]	0x[illegible]000	0x1000	0x40a000	0xa5a5000	0x1000
0x4080[illegible]	[illegible]0ec000	0x1000	0x40b000	0x1fa26000	0x1000
0x409000	0xf0ed000	0x1000	0x40c000	0x1b610000	0x1000
			0x40e000	0x9fd1000	0x1000
			0x40f000	0x1da52000	0x1000

[illegible] of memory map of the process "hedef100.exe" and "user_load.exe"

The VAD tree describes memory ranges is used by a process and enables reconstruction of a process virtual address space. A node in this tree can have a number of different pool tags, depending on the type of Virtual Address Descriptor. Common tags are VadS, Vad and VadL. The latter two objects contain pointers to Control Areas, which are described below. This object contains, among other things, the file size of the mapped file. Fig. 2 illustrates the virtual address, physical address and the page size. In each process, An essential part of the operating system on the suspect machine and distinguishing legitimate components from suspicious and potentially malicious applications.

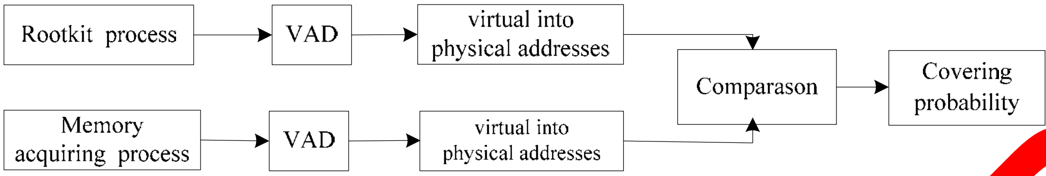

Fig.3 The flow char of the computing the key trace covering probability

During the process, we employ volatility framework 2.0 to analyze the memory[12]. Fig.3 illustrates the flow chart of the computing the key trace covering probability. Firstly, we capture the whole memory of the target system. Later, we search the rootkit process, and find the Pid. According to the Pid, we find out the virtual address using VAD tools. Then, we translate the virtual address into physical address and record them. At last, we compare the physical address employed in the rootkit with that employed in the memory acquired tool. In the real running system, there are some rootkits, such as hxdef100.exe. Usually, we kill the process using icesword tool, and capture the system memory once again.

4. Experiments and results

In order to make clear which part of memory content has been changed during the evidence acquired process. And which part of memory content would be covered because of loading the memory acquired tool, we make some experiments using the following Environment. The experiments are performed using a Windows XP Service Pack 2 VMware® Workstation 7.1.4 [illegible]5536 on a Windows 7 host. The host OS is Windows 7 Ultimate, 32-bit (Build [illegible]00) 7600, with 3G RAM. The live response toolkit used in our experiments is the user_load.exe, which is developed by ourselves and based on the absolute driver file (MemDump.sys).

The comparing files function of the WinHex, using to analysis the impact on progress and drive of the key traces, by comparing the front and back memory image files of running forensics tools. A txt file will be generated if they are different, [illegible] difference was found, so forensics tools have no effect on the progress and drive of Hacker Defender(hxdef100.exe).

The size of the Hxdef100.exe occupied 292KB on disk. However, the pages of hxdef100.exe occupied 136 in memory. [illegible] x nt! M[illegible]ailablepages command using Windbg to find the basic addresses of the unallocated memory pages, moreover operating MemoryAnalyzer to obtain the offset address. Finally, get the physical address of the undistributed memory pages by adding the above two addresses. We make different experiments using different rootkits, such as hack defender. Running rookits also need some shared Dlls, and this Dlls memory is shared. Here, we only consider the part of .exe memory.

Tab.1 the probability of covering key trace

	[illegible]	m	n	m-n	$P(A_i)$
Hxdef100.exe	136	0x015ad2(88786)	0x0159ef(88559)	227	0.5991

Before acquiring the system memory, the unallocated pages are 88786, and after acquiring the memory, the unallocated pages are 88559. Therefore, the new allocated memory page is 88786-88559=227. The pages of the Hxdef100 process occupied 136. In theory, the probability of covering key trace is 0.5991. From formula (3) in section 3.2, we can compute the covering probability 1-(1-0.5991)^136 is nearly to 1. That is, the integrity of the process hxdef100.exe is destroyed.

From the process physical memory, about 49 in 134 pages are the same, that is, 49 pages are reused in the process of the memory acquired tool. It can be noted that the hxdef100.exe contains 136pages, but 134 pages can be located in the physical memory. The results of Tab.1 show that part of the key trace is covered by loading the memory acquired tool. That is, the integrality of the original data using in the malware possibility in the memory can be destroyed. Some of physical memory employed in the original process may be allocated for running the new process.

5. Conclusions and future work

When the process has just been killed, not all the physical memory related the process wo[illegible] be allocated for other process. In this paper, we analyze the allocated physical memory of the [illegible] trace. Then, we kill the process and count the covering pages. Considering the relation between usage [illegible] of memory and model of the key trace, we computed the covering probability in the capture[illegible]emory. S[illegible] of the data in process can not be very important in court. Therefore, we need further t[illegible]vestig[illegible] that t[illegible] key data in the process would be covered or not in detail. In the future work, we wil[illegible]e th[illegible]roblem.

Acknowledgements

This work is supported by grants by National Natural Science F[illegible]ndati[illegible]f China (Grant No. 61070163), by the Shandong Province Outstanding Research Award F[illegible]d for You[illegible] Scientists of China (Grant No. BS2011DX034) and by the Natural Science Founda[illegible] of Shandong Pro[illegible]ce, China(Grant No.ZR2009GM036).

References

[1] Stefan V, Felix C. Freiling, A survey of main memory [illegible]uisiti[illegible]alysis techniques for the windows operating system , *digital investigation* 8(2011)3-22.

[2] James Okolica, Gilbert L. Peterson, Ex[illegible]e wind[illegible] clipboard from physical memory, *digital investigation* 8(2011)S118-S124

[3] Stevens R, Casey E. Extracting W[illegible]ws co[illegible] Det[illegible]ls from Physical memory. In: *Proceedings of the 2010 digital forensic research Workshop (DF[illegible]S)*; 2[illegible]e63.

[4] Brendan Dolan-Gavitt, Fo[illegible] analysis o[illegible] Windows registry in memory, *Digital Investigation* 5 (2008)S26-S32

[5] Shuhui Z, Lianhai W, [illegible]xtracting win[illegible] registry information from physical memory, *2011 3rd International Conference on Computer Resea[illegible]h and D[illegible]pment (ICCRD)*,Issue Date: 11-13 March 2011 On page(s): 85 - 89.

[6] Walters A, Pet[illegible] ,Jr NL. Volat[illegible] integrating volatile memory forensics into the digital investigation process. In: *Black Hat DC 2007;* [illegible].

[7] Lianhai [illegible]uicha[illegible] Shuhui Z, A Model of Computer Live Forensics Based on Physical Memory Analysis. *The 1st International Confer[illegible]formatio[illegible]ience and Engineering*, pp. 4647-4649

[8] [illegible] Z, L[illegible] W, S[illegible]ui Z, Windows Memory Analysis Based on KPCR, ias, vol. 2, pp.677-680, *2009 Fifth Intern[illegible]al Co[illegible]ence on [illegible]tion Assurance and Security*, 2009

[9] Brend[illegible] Dolan-Gavitt, The VAD tree: A process-eye view of physical memory, *digital investigation* 4(2007)S62-S64

[10] [illegible]nk W, van A.R. Ballegooij, Forensic memory analysis: Files mapped in memory, *digital investigation* 5(2008)S[illegible]57.

[11] Ru[illegible]vich ME, Solomon DA, Ionescu A. Microsoft windows internals. 5th ed. *Microsoft Press*; June 2009.

[12] The [illegible] Volatility Framework: Volatile memory artifact extraction utility framework www.volatilesystems.com/default/volatilty,