Extracting Text from Windows XP Memory Image

Long Chen[1,2], Lei Kang[1], Zhenxing Dong[1,2]

(1. Institute of Computer Forensics, Chongqing University of Posts and Telecommunications, Chongqing 400065,P. R. China；2. Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, P. R. China

chenlong@cqupt.edu.cn

Abstract: User data from physical memory contains lots of important information that are of potential evidential value in forensic analysis. This article analyzes and summarizes the rules of text files stored in the notepad’s process in Windows XP operating system, and then proposes a method of extracting text from physical memory image. The experimental results show that the method can accurately extract the text from image which contains the living notepad’s process.

Keywords: memory forensics; text file; physical memory analysis; digital forensics

1. Introduction

There has been significant research about system structures into memory forensics. In 2006, Schuster A [1] proposed a method for extracting processes and threads information, and this method has been proven successful in identifying system objects that are associated with closed or disconnected processes, files and connections. In 2008, Dolan-Gavitt B [2] analyzed the structure of the registry in memory, and extracted the registry to identify whether the system had been attacked. In 2009, based on a data structure in windows known as Kernel Processor Control Region, Ruichao Zhang and Lianhai Wang [3] proposed a new method to extract _EPROCESS from memory image. In 2010, Okolica J and Peterson G[4] created a flexible tool (CAMT) using the debug structures embedded in memory images and Microsoft’s program database (PDB) files, and the tool can take an arbitrary memory image from any of the family of Windows NT operating systems and extract process, configuration, and network activity information.

There is a need for more memory forensic techniques to extract user data retained in various Microsoft Windows applications. In 2010, Stevens R and Casey E [5] focused on finding and carving unique signatures for command line history in memory images without tracking it back to the originating processes. In 2011, Okolica J and Peterson G described the structure of the Windows clipboard [6], and put forward a method to retrieve the text in clipboard.

In addition to command line and clipboard, text file are also critical to forensic analysis. Since Windows1.0 was published in 1985, all versions of Microsoft Windows embedded the text edit tool notepad. Therefore, if we can reconstruct the contents of the text file from notepad’s process, it might provide important and plain clues for investigation.

This paper describes the rule how the text file data are stored in memory, and then proposes a method to extract the text from physical memory image, where the text is located in living notepad’s process.

2. Methodology

2.1 Rule

In order to understand more fully the functionality of the Notepad application and how it stores the text in memory, it was necessary to conduct a number of live experiments. Initial experiments were performed on Windows XP Professional SP3 (VOL) on VMware Workstation 8.0.1 configured with 512MB of RAM，and the physical address expansions is disabled.

When text file is loaded into memory, the physical addresses in memory are not continuous usually, but its virtual addresses in Notepad's process are continuous.

Experiments revealed that text in memory is in Unicode，and a 12 bytes flag"0*x*010004002e 00740078007400" exists in front of the starting position of text. Moreover, the virtual addresses of text in Notepad's process are affected by its size.

The number of characters of the text is stored with four bytes in the virtual page 0*x*aa, and can be located with a flag "0*x*ffffffff". Because text in memory is encoded in Unicode, Each Chinese or English character is represented with two bytes. Set d as the number of character in memory, and set m as the size of the text in memory, then $m=2d$.

With the size of text opened increasing, both SVPN (starting virtual page number) and EVPN (ending virtual page number) of the text in notepad's process move forward. The process of change is shown in Fig. 1.

Size	Smaller ←——————————————————————→ bigger												
SVPN	af	af	b0	b0	b0	b1	b1	b1	b2	b2	b2	b3	b3	...
EVPN	af	b0	b0	b1	b2	b2	b3	b4	b4	b5	b6	b6	b7	…

Fig.1 change process of virtual page number

The starting position of a text in notepad's process is an offset from the beginning of a page, and the size of page is 4KB. The interval of virtual page number that text occupies in process can be computed by its size. For example, if the size of text in memory is not greater than 4KB, then the text must be provided with one page or two pages in memory. In this case, we can see from Fig. 1, the virtual page number of the text may be 0xaf, 0*x*b0, 0*x*af and 0*x*b0, or 0*x*b0 and 0*x*b1. So, the virtual page number of the text must be in an interval, i.e. between 0*x*af and 0*x*b1. Similarly, if the size of text in memory is greater than 4KB but not greater than 8KB, the virtual page number of the text must be between 0*x*b0 and 0*x*b3.

Set [a, b] as an interval to include all the virtual page number of the text. A rule to compute a and b is formulated as follows (Symbol " $\lfloor \mathrm{X} \rfloor$ " defined that takes the smallest integer which is greater than or equal to the number "X"):

$$a = 0x\mathrm{ae} + \lfloor m / 4\mathrm{KB} \rfloor \quad (1)$$

$$b = 0x\mathrm{af} + \lfloor m / 4\mathrm{KB} \rfloor * 2 \quad (2)$$

2.2 Address Translation

Virtual addresses needs to be translated into physical addresses to be able to locate the text in memory image. Page directory base address which is necessary for address translation [7] is stored in the EPROCESS Structure. The address of _EPROCESS structure of Notepad process in physical memory can be located through searching strings, the page directory base address at the offset 0*x*18. The address translation process is shown in Fig 2.

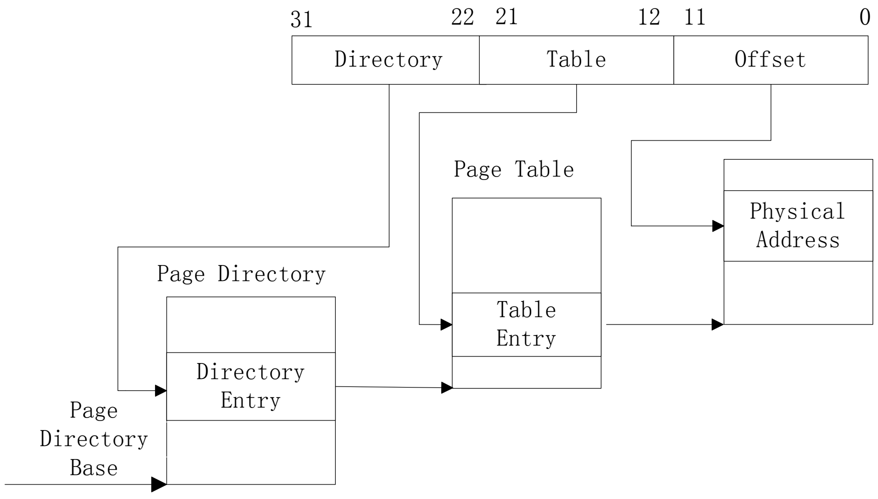


Fig.2 Address Translation

Based on the rules and the principle of address translation, the process of extracting text is as follows:

(1) The address of _EPROCESS structure of Notepad process in physical memory is located through searching strings, and then gets the page directory base address.

(2) Search the sign "0*x*ffffffff" in the virtual page 0*x*aa, and obtain the number of characters *d*, and then get the size of the text in memory.

(3) The virtual page number interval is computed based on the rule of text size in memory.

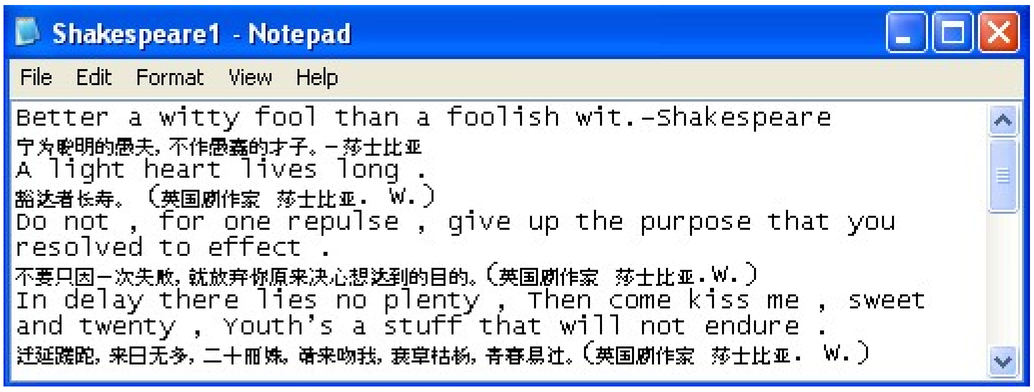
(4) Physical address of pages in the interval is translated from virtual address by page directory base address.

(5) Search for the flag of text starting position in the interval data, and then according to the size *m* write the contents into a file.

3. Results and analysis

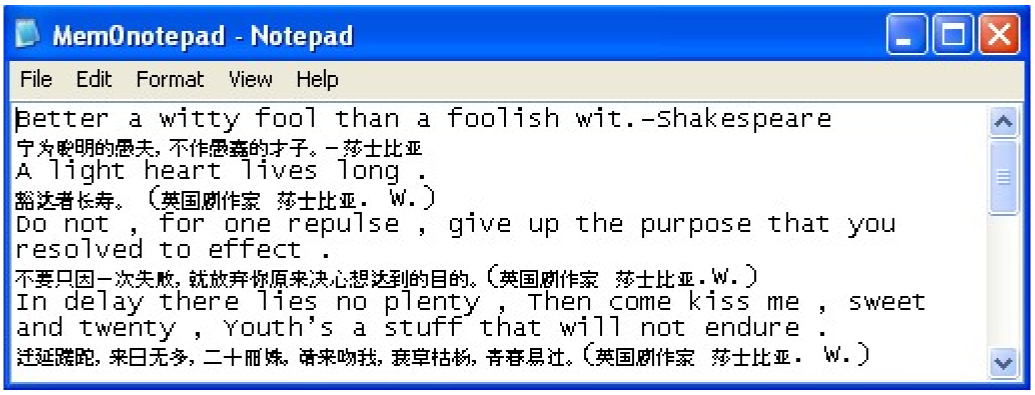
Eighty memory images are created for analysis by DumpIt[8], and every image is created on Lenovo laptop with Windows XP (SP3 VOL) which physical address extensions is disabled, and the size of memory is 512M.

A tool is implemented with above rules. The tool can extract text from memory image correctly for all images. One example is shown in Fig. 3 and Fig. 4.



```
Better a witty fool than a foolish wit.-Shakespeare
宁为聪明的愚夫，不作愚蠢的才子。－莎士比亚
A light heart lives long .
豁达者长寿。 （英国剧作家 莎士比亚． W．）
Do not , for one repulse , give up the purpose that you
resolved to effect .
不要只因一次失败，就放弃你原来决心想达到的目的。（英国剧作家 莎士比亚．W．）
In delay there lies no plenty , Then come kiss me , sweet
and twenty , Youth's a stuff that will not endure .
迁延蹉跎，来日无多，二十丽姝，请来吻我，衰草枯杨，青春易逝。（英国剧作家 莎士比亚． W．）
```

Fig. 3 Original File



```
Better a witty fool than a foolish wit.-Shakespeare
宁为聪明的愚夫，不作愚蠢的才子。－莎士比亚
A light heart lives long .
豁达者长寿。 （英国剧作家 莎士比亚． W．）
Do not , for one repulse , give up the purpose that you
resolved to effect .
不要只因一次失败，就放弃你原来决心想达到的目的。（英国剧作家 莎士比亚．W．）
In delay there lies no plenty , Then come kiss me , sweet
and twenty , Youth's a stuff that will not endure .
迁延蹉跎，来日无多，二十丽姝，请来吻我，衰草枯杨，青春易逝。（英国剧作家 莎士比亚． W．）
```

Fig. 4 Generated File

4. Conclusion

This paper described some rules of text in memory, and proposed a method to locate the virtual space of text in memory. We developed a tool to restore the text in memory through address translation. Experimental results show that the tool developed can recover text file correctly.

Preliminary results show that the number of starting virtual page may be different in other versions of Windows XP in general, but these rules are also suitable to them. So the corresponding changes should be done in the rule mentioned in section 2.1

Acknowledgements

This work is supported by Science & Technology Research Program of the Municipal Education Committee of Chongqing of P. R. China (No.KJ110505), Key Project of Science and Technology and Natural Science Foundation Project of CQ CSTC of P. R. China (No. cstc2011AC2155, No. cstc2011jjA40031).

REFERENCES

[1] Schuster A. Searching for processes and threads in Microsoft Windows memory images. In: Proceedings of the 2006 digital forensic research Workshop (DFRWS); 2006. p. 6-10.

[2] Dolan-Gavitt B. Forensic analysis of the Windows registry in memory, Proceedings of the 2008 digital forensic research Workshop (DFRWS); 2008. p. 26-32.

[3] Ruichao Zhang, Lianhai Wang, Shuhui Zhang. Windows memory analysis based on KPCR. IAS, 2009 Fifth international conference on information assurance and security; 2009 .vol.

2, p.677-680,

[4] Okolica J, Peterson G. Windows operating systems agnostic memory analysis. In: Proceedings of the 2010 digital forensic research Workshop (DFRWS); 2010. p. 48-56

[5] Stevens R, Casey E. Extracting Windows command line details from physical memory. In: Proceedings of the 2010 digital forensic research Workshop (DFRWS); 2010. p. 57-63.

[6] Okolica J, Peterson G. Extracting the windows clipboard from physical memory. In: Proceedings of the 2011 digital forensic research Workshop (DFRWS); 2011. p. 48-56.

[7] Intel Corporation. Intel 64 and IA-32 architectures software developer’s manual volume 3A: System programming guide [EB/OL]. (2007—11). Http: //www. inte1.corn/design/ processor/manuals/253668.pdf.

[8] MoonSols. Windows memory " DumpIt v1.3.2.20110401". http://www.moons- ols.com.