Acquisition of Network Connection Status Information from Physical Memory on Windows Vista Operating System

Lijuan Xu, Lianhai Wang, Lei Zhang, and Zhigang Kong

Shandong Provincial Key Laboratory of Computer Network,
Shandong Computer Science Center
19 Keyuan Road, Jinan 250014, P.R. China
{xulj,wanglh,zhanglei,kongzhig}@keylab.net

Abstract. A method to extract information of network connection status information from physical memory on Windows Vista operating system is proposed. Using this method, a forensic examiner can extract accurately the information of current TCP/IP network connection information, including IDs of processes which established connections, establishing time, local address, local port, remote address, remote port, etc., from a physical memory on Windows Vista operating system. This method is reliable and efficient. It is verified on Windows Vista, Windows Vista SP1, Windows Vista SP2.

Keywords: computer forensic, memory analysis, network connection status information.

1 Introduction

In living forensics, network connection status information describes computer's activity communicating with outside world when the computer is investigated. It is important digital evidence judging whether respondents are doing illegal network activity or not. As a volatile data, current network connection status information exist in physical memory of a computer[1]. Therefore, acquiring this digital evidence depends on analyzing physical memory of the computer.

There are a number of memory analysis tools, for examples, WMFT(Windows Memory Forensic Toolkit), volatools, memparser, PTFinder, FTK, etc. WMFT[2] can be used to perform forensic analysis of physical memory images acquired from Windows 2000/2003/XP machines. PTFinder(Process and Thread Finder) is a Perl script created by Andreas Schuster[3] to detect and list all the processes and threads in a memory dump. MemParser tool was programmed by Chris Betz which can enumerate active processes and could also dump their process memory[4]. volatools[5] is a commandline toolkit intended to assist with the Survey Phase of a digital investigation, it is focused on support for Windows XP SP2 and can collect open connections and open ports which could typically be obtained by running netstat on the system under investigation[6,7,8].

X. Lai et al. (Eds.): E-Forensics 2010, LNICST 56, pp. 122–130, 2011.
© Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2011

Windows Vista is the new Microsoft operating system that was released to the public at the beginning of 2007. There are many changes to the new Windows Vista operating system compared to previous versions of Microsoft Windows that has brought new challenges for digital investigations. The tools metioned aboved can not acquire network connection status information from Windows Vista operating system. A memthod to extract network connection status information from physical memory on Windows Vista operating system is not published so far.

2 Related Work

Nowadays, there are two methods to acquire network connection status information from physical memory of Windows XP operating system. One is searching for data structure "AddrObjTable" and "ObjTable" from driver "tcpip.sys" to acquire network connection status information. This method is implemented in Volatility[9], a tool to analyze memory which dumps from Windows XP SP2 or Windows XP SP3 for an incident response perpective developed by Walters and Petroni. The other one is proposed by Schuster[10]. Schuster descirbes the steps necessary to detect traces of network activity in a memory dump. His method is searching for pool allocations labeled "TCPA" and a size of 368 bytes (360 bytes for the payload and 8 for the _POOL_HEADER) on Windows XP SP2. These allocations will reside in the non-paged pool.

The first method is feasible on Windows XP. It doesn't work on Windows Vista, because there is no data structure "AddrObjTable" or "ObjTable" in driver "tcpip.sys". It is proven that there is no pool allocations labeled "TCPA" on Windows Vista as well.

It is analyzed that there are pool allocations labeled "TCPE" instead of "TCPA" indicating network activity in a memory dump of Windows Vista. Therefore, we can acquire network connections from pool allocations labeled "TCPE" on Windows Vista.

This paper proposes a method of acquiring current network connection informations from physical memory image of Windows Vista according to memory pool. Network connection information including IDs of processes which established connections, establishing time, local address, local port, remote address, remote port, etc., can be get accurately from physical memory image file of Windows Vista with this method.

3 Acquisition of Network Connection Status Information from Physical Memory on Windows Vista Operating System

A method of acquiring current network connection information from physical memory image of Windows Vista based on memory pool is proposed.

3.1 The Structure of TcpEndpointPool

A data structure called TcpEndpointPool is found in driver "tcpip.sys" on Windows Vista operating system. This pool is a doubly-linked list of which each node is the head of a singly-linked list.

The internal organizational structure of TcpEndpointPool is shown by figure1. The circles represent heads of the singly-linked list. The letters in the circles represent the flag of the head. The rectangles represent the nodes of singly-linked list. The letters in the rectangles represent the type of the node.

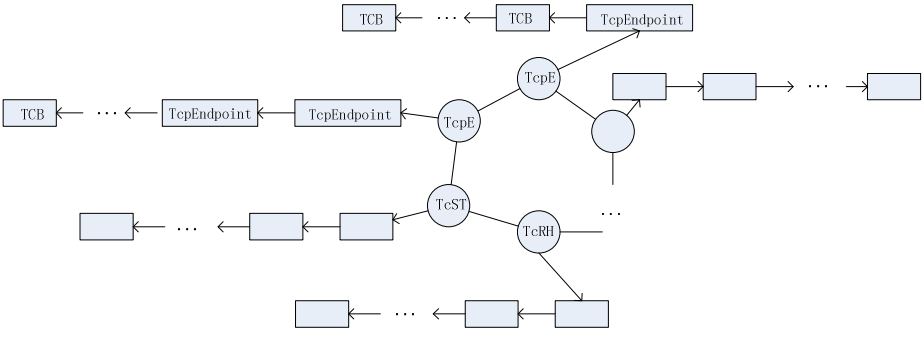


Fig. 1. TcpEndpointPool internal organization

The structure of singly-linked list head is shown by figure 2, in which there is a _LIST_ENTRY structure at the offset 0x30 by which the next head of a singly-linked list can be found.

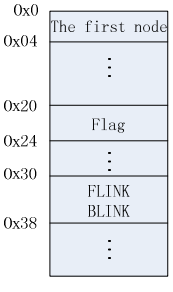


Fig. 2. The structure of singly-linked list head

The relationship of two adjacent heads is shown by figure 3.

There is a flag at the offset 0x20 of the singly-linked list head by which the node structure of the singly-linked list can be judged. If the flag is "TcpE", the singly-linked list with this head is composed of TcpEndPoint structure and TCB structure which describe the network connection information.

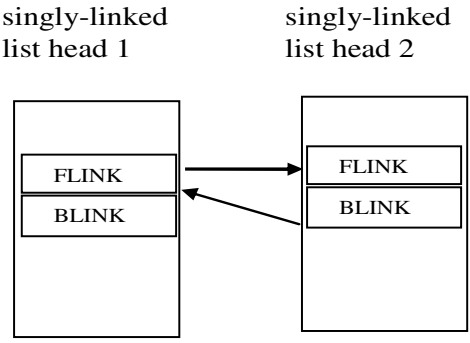


Fig. 3. The linked relationship of two heads

3.2 Searching for TcpEndpointPool

The offset of TcpEndpointPool's address relative to the base address of tcpip.sys is 0xd0d5c for Windows Vista SP1 and 0xd3e9c for Windows Vista SP2. Therefore, the virtual address of TcpEndpointPool can be computed by 0xd0d5c adding the virtual address of tcpip.sys's base address for Windows Vista SP1 and 0xd3e9c adding the virtual address of tcpip.sys's base address for Windows Vista SP2.

The base address of driver tcpip.sys can be acquried by using PsLoadedModuleList which is a global variable. That is because PsLoadedModuleList is a pointer to the list of currently loaded kernel modules, the base address of all loaded drivers can be acquried according to this variable.

3.3 TcpEndpoint and TCB

The definition and the offsets of fields related with network connections in the TcpEndPoint structure is shown as follows.

```
typedef struct _TCP_ENDPOINT {
    PEPROCESS OwningProcess;                    +0x14
    PETHREAD OwningThread;                      +0x18
    LARGE_INTEGER CreationTime;                 +0x20
    CONST NL_LOCAL_ADDRESS* LocalAddress;       +0x34
    USHORT LocalPort;                           +0x3e
} TCP_ENDPOINT, *PTCP_ENDPOINT;
```

From above structure, a pointer points to the process which established network connections at the offset 0x14, and a pointer points to the thread which established network connections at the offset 0x18.

The definition and the offsets of fields related with network connection information in the Tcb structure is shown as follows.

```
typedef struct _TCB {
    CONST NL_PATH *Path;                    +0x10
    USHORT LocalPort;                       +0x2c
    USHORT RemotePort;                      +0x2e
    PEPROCESS OwningProcess;                +0x164
    LARGE_INTEGER CreationTime;             +0x16c
} TCB,  *PTCB;
```

NL_PATH structure, NL_LOCAL_ADDRESS structure and NL_ADDRESS_IDENTIFIER structure are defined as follows by which network connection local address and remote address can be acquried.

```
typedef struct _NL_PATH {
    CONST NL_LOCAL_ADDRESS *SourceAddress;      +0x00
    CONST UCHAR *DestinationAddress;            +0x08
} NL_PATH, *PNL_PATH;
typedef struct _NL_LOCAL_ADDRESS {
    CONST NL_ADDRESS_IDENTIFIER *Identifier;    +0x0c
} NL_LOCAL_ADDRESS, *PNL_LOCAL_ADDRESS;
typedef struct _NL_ADDRESS_IDENTIFIER {
    CONST UCHAR *Address;                       +0x00
} NL_ADDRESS_IDENTIFIER, *PNL_ADDRESS_IDENTIFIER;
```

Comparing the definition of TCP_ENDPOINT structure with the definition of TCB structure, we can say that if a pointer points to a EPROCESS structure at the offset 0x14 of the structure (the first 4 bytes of EPROCESS structure is 0x3002000 for windows Vista operating system), this structure is TCP_ENDPOINT, otherwise this structure is TCB.

4 Algorithm

4.1 The Overall Algorithm of Extracting Network Connection Information

The overall flow of extracting network connection information for Windows Vista operating system is shown by figure 4.

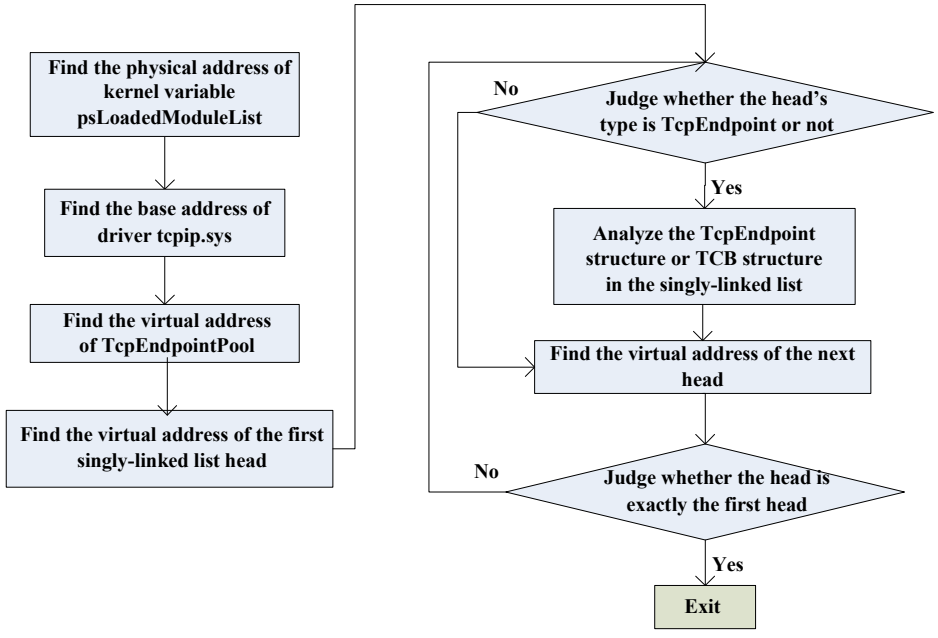


Fig. 4. The flow of extracting network connection information for Windows Vista operating system summary description

The algorithm is given as follows.

Step1 Get the physical address of kernel variable psLoadedModuleList using windows memory analyzing method based on KPCR[11].

Step2 Find the base address of driver tcpip.sys according to physical address of PsLoadedModuleList which point to a doubly-linked list composed of all drivers in the system.

Step3 Find the virtual address of TcpEndpointPool.

Step4 Find the virtual address of the first singly-linked list head.

Firstly, transfer the virtual address of TcpEndpointPool to physical address and locate the address in the memory image file. Secondly, read 4 bytes at this position and transfer the 4 bytes to physical address, locate the address in the memory image file. Lastly, the virtual address of the first singly-linked list head is the 4 bytes at the offset 0x1c.

Step5 Judge whether the head's type is TcpEndpoint or not by reading the flag which is set at the offset 0x20 relative to the head's address. If the flag is "TcpE", the head's type is TcpEndpoint, go to the step 6, otherwise go to the step 7.

Step6 Analyze the TcpEndpoint structure or TCB structure in the singly-linked list. Analyzing algorithm is shown by figure 5.

Step7 Find the virtual address of the next head.

The virtual address of the next head can be found according to the _LIST_ENTRY structure which is set at the offset 0x30 relative to the address of singly-linked list head. Judging whether the next head's virtual address equals to the first head's address or not. If the next head's virtual address is equal to the first head's address, exit the procedure, otherwise go to the next step.

Step8 Judge whether the head is exactly the first head. If the head is exactly the first head, exit, otherwise go to step 5.

The flow of analyzing TCB structure or TcpEndpoint structure is shown as follows.

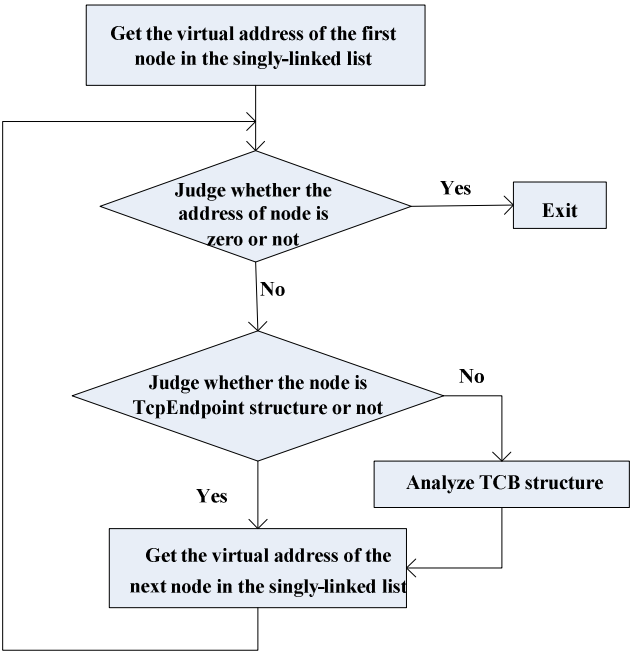


Fig. 5. The flow of analyzing TCB structure or TcpEndpoint structure summary description

Step1 Get the virtual address of the first node in the singly-linked list.

Transfer the virtual address of singly-list head to physical address and locate the address in memory image file. Read 4 bytes from this position which is the virtual address of the first node.

Step2 Judge whether the address of node is zero or not. If the address is zero, exit the procedure, otherwise go to the next step.

Step3 Judge whether the node is TcpEndpoint structure or not.

Transfer the virtual address of the ndoe to physical address and locate the address in the memory image file. Put 0x180 bytes from this position into a buffer. Read 4 bytes at buffer's offset 0x14 and judge whether the value is a pointer which point to a

EPROCESS structure or not. If the value is a pointer which point to a EPROCESS structure, go to step 5, otherwise it indicates that the node's structure is TCB structure, go to the next step.

Step4 Analyze TCB structure.

Step4.1 Get PID (process id) which is the ID of the process which established this connection. The pointer which points to the process's EPROCESS structure which established this connection is set at the offset 0x164 relative to TCB structure. Firstly, read 4 bytes which represents the virtual address of EPROCESS structure at buffer's offset 0x164 and transfer it to physical address. Secondly, locate the address in the memory image file and read 4 bytes which represents PID at the offset 0x9c relative to EPROCESS structure's physical address.

Step4.2 Get establishing time of this connection. The number is set at the offset 0x16c of TCB structure . Read 8 bytes at offset 0x16c of the buffer and it represents establishing time.

Step4.3 Get the local port of this connection. The number is set at offset 0x2c of TCB structure. Read 2 bytes at offset 0x2c of the buffer and transfer it to a decimal which is the local port of this connection.

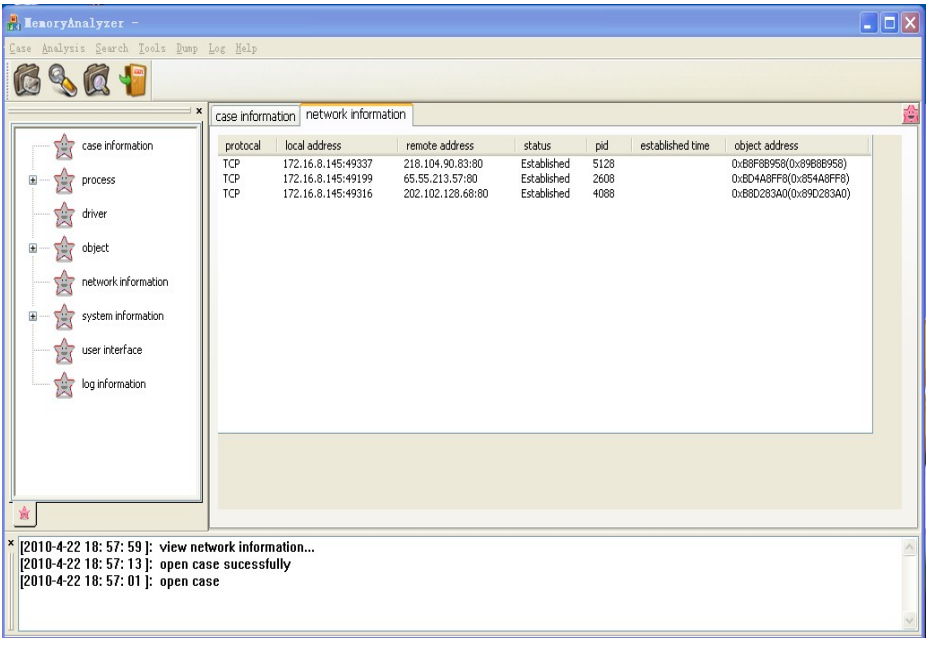
Step4.4 Get the remote port of this connection. The number is set at the offset 0x2e of TCB structure. Read 2 bytes at offset 0x2e of the buffer and transfer it to a decimal which is the remote port of this connection.

Step4.5 Get local address and remote address of this connection. The pointer which points to NL_PATH structure is set at the offset 0x10 of TCB structure. The pointer which points to the remote address is set at the offset 0x08 of NL_PATH structure. The special algorithm is as followes: read 4 bytes which represents the virtual address of NL_PATH structure at the offset 0x10 of TCB structure, transfer the virtual address of NL_PATH structure to physical address, locate the address+0x08 in the memory image file and read 4 bytes which represents remote address at this position. The pointer which points to NL_LOCAL_ADDRESS structure is set at the offset 0x0 of the TCB structure, The pointer which points to NL_ADDRESS_IDENTIFIER structure is set at the offset 0x0c of TCB structure, local address is set at the offset 0x0 of the NL_ADDRESS_IDENTIFIER structure. Therefore, local address can be acquired from the above three structures.

Step5 Get 4 bytes which represents the next node's virtual at the offset 0 of the buffer and go to step2.

5 Conclusion

In this paper, a method which can acquire network connection information on Windows Vista operating system memory image file based on memory pool allocation strategy is proposed. This method is reliable and efficient, because the data structure TcpEndpointPool exists in driver tcpip.sys for every Windows Vista operation system version and TcpEndpointPool structure will not change when Windows Vista operation system version changed. A software which implements this method is present as follows.



References

1. Brezinski, D., Killalea, T.: Guidelines for evidence collection and archiving. RFC 3227 (Best Current Practice) (February 2002), http://www.ietf.org/rfc/rfc3227.txt
2. Burdach, M.: Digital forensics of the physical memory, http://forensic.seccure.net/pdf/mburdachdigitalforensicsofphysicalmemory.pdf
3. Schuster, A.: Searching for processes and threads in Microsoft Windows memory dumps. Digital Investigation 3(supplement 1), 10–16 (2006)
4. Betz, C.: memparser, http://www.dfrws.org/2005/challenge/memparser.shtml
5. Walters, A., Petronic, N.: Volatools: integrating volatile memory forensics into the digital investigation process. Black Hat DC 2007 (2007)
6. Jones, K.J., Bejtlich, R., Rose, C.W.: Real Digital Forensics. Addison Wesley, Reading (2005)
7. Carvey, H.: Windows Froensics and Incident Recovery. Addison Wesley, Reading (2005)
8. Mandia, K., Prosise, C., Pepe, M.: Incident Response and Computer Forensics. McGrawHill Osborne Media (2003)
9. The Volatility Framework: Volatile memory artifact extraction utility framework, https://www.volatilesystems.com/default/volatility/
10. Schuster, S.: Pool allocations as an information source in windows memory forensics. In: Oliver, G., Dirk, S., Sandra, F., Hardo, H., Detlef, G., Jens, N. (eds.) IT-Incident Management & IT-Forensics-IMF 2006. Lecture notes in informatics, vol. P-97, pp. 104–115 (2006)
11. Zhang, R.C., Wang, L.H., Zhang, S.H.: Windows Memory Analysis Based on KPCR. In: 2009 Fifth International Conference on Information Assurance and Security, IAS, vol. 2, pp. 677–680 (2009)