A Novel File Carving Algorithm for EVTX Logs

Ming Xu[1,2(✉)], Jinkai Sun[1], Ning Zheng[1], Tong Qiao[2], Yiming Wu[2], Kai Shi[1], Haidong Ge[1], and Tao Yang[3(✉)]

[1] Internet and Network Security Laboratory, School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
{mxu,152050160,nzheng,12084232,151050149}@hdu.edu.cn
[2] School of Cyberspace, Hangzhou Dianzi University, Hangzhou, China
{tong.qiao,ymwu}@hdu.edu.cn
[3] Key Lab of the Third Research Institute of the Ministry of Public Security, Shanghai, China
yangtao@stars.org.cn

Abstract. The Microsoft Windows system provides very important sources of forensic evidence. However, few attention has been paid to the recovery of the deleted EVTX logs. Without using system metadata, a novel carving algorithm of EVTX logs is proposed by analyzing the characteristics and intrinsic structure. Firstly, we reassemble binary data belonging to fragments of complete EVTX logs to reconstruct the deleted logs. Secondly, extracting records for the corrupted logs can make the algorithm robust through the special features of template and substitution array. Finally, some experiments are given to illustrate the effectiveness of the proposed algorithm. Moreover, when the logs are fragmented or corrupted, our algorithm can still perform well.

Keywords: Windows forensics · Windows XML event logs
EVTX Files · File carving · Fragmented files

1 Introduction

Since log files generally link a certain event to the special time, they can provide very important sources of forensic investigation. It is very easy for an internal employee to steal or destroy the information of the company computers. During committing illegal activities, a criminal possibly removes or hides traces after his crime behavior. It makes operations untraceable with no digital evidence left. Therefore, the technique which can help us to recover maliciously deleted logs has received significant attention over the past few years [1].

As a replacement for the Windows event log (EVT) format, the Windows XML event log (EVTX) format was first introduced in Vista for less storage through binary XML technology. EVTX logs provide a great deal of basic and valuable information such as name of the account, created time, record number

P. Matoušek and M. Schmiedecker (Eds.): ICDF2C 2017, LNICST 216, pp. 97–105, 2018.
https://doi.org/10.1007/978-3-319-73697-6_7

and event ID which could be used to identify the specific kind of an event. For instance, event ID 4624 means that an account was logged on. It is confirmed that a criminal logged on a computer at a certain time associating with the included time and username.

Nevertheless, criminals are always expected to conceal their criminal records by deleting logs. Because of file fragmentation on actual file systems [2], it is too time consuming to use a brute-force approach dealing with each possible order without file system information. Thus we present a novel carving algorithm to extract deleted records and demonstrate the effectiveness of our proposed algorithm by comparing it with the commercial forensic software Encase[1].

2 Related Work

Several researchers have noted that logs of Windows contain a large amount of useful digital evidence [3,4]. Schuster first provides description about the newer EVTX format [5], and XML technology is adopted to parse Vista event log files [6]. For different Windows systems, Windows 8 event log format is introduced [7]. In addition, Do et al. present a Windows event forensic process for analyzing log files [8].

Moreover, researchers focus on caving contiguous files firstly [9,10]. For fragmented files, some carving algorithms based on file signature are proposed [11,12] and a novel framework is designed to resolve this problem [2,13]. Unfortunately, there has been relatively few papers published for file carving of the EVTX logs. Therefore, in this context, we propose a novel file carving algorithm to deal with this challenge.

3 Description of EVTX Logs

By investigating the characteristics and internal structure of EVTX Logs (see Fig. 1), we can smoothly establish our algorithm for realizing forensics.

3.1 File Header

Each log file contains a file header, which describes the basic information of the file. A file header occupies 4096 bytes space which is a complete cluster, but uses only 128 bytes actually. In our algorithm, the checksum which verifies integrity of the file header is gained through the CRC32 (Cyclic Redundancy Check) method to calculate the first 120 bytes of the file header. We use *magic string* "ElfFile" and *checksum* to find a integrated file header for marking the following chunk as the first chunk of the file.

[1] EnCase offers investigators the flexibility to collect critical evidence including text messages, call records, pictures, graphics, and much more.

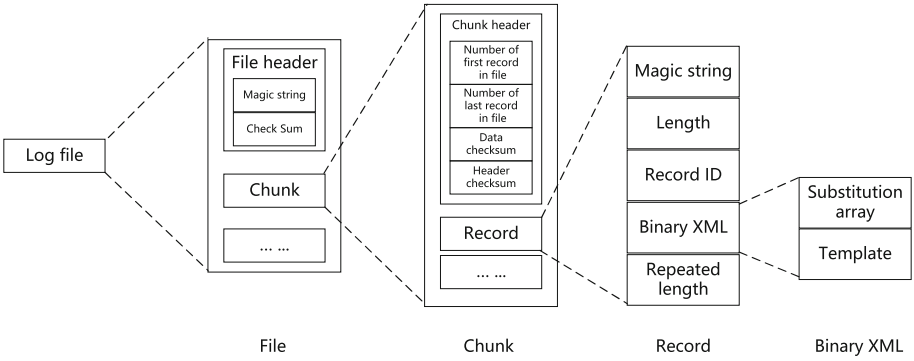


Fig. 1. File structure

3.2 Chunk Header

Each chunk consists of a smaller header and a series of event records. It starts with the *magic string* "ElfChnk", which helps to identify the chunk. Chunk header provides two different sets of counters for *record ID*[2], and for the same chunk it is safe to assume that *record ID* of the included record is in the range between *number of first record in file* and *number of last record in file*. It can contribute to determining whether a record belongs to the original chunk by the information of the chunk header.

Checksum is important for guaranteeing the integrity of the chunk. *Data checksum* is calculated for the CRC32 of all the records data belonging to this chunk. In addition, *header checksum* is the CRC32 of the first 120 bytes and bytes 128 to 512 of the chunk header. Therefore, we can use *header checksum* to confirm the integrity of the chunk header and *data checksum* to check whether the records of chunk are found completely.

3.3 Record

Each event record contains basic information. A fragment belongs to a record potentially for the existence of the *magic string* "**". *Length* and *repeated length* allow us to find a complete record.

The main content of the record is coded through the binary XML technology. Binary XML mainly involves two concepts: template and substitution array. Binary XML starts with a template which is transformed from a sequence of tokens and a template has some substitution tokens which are needed to be filled with the value of substitution array (see Fig. 2). Template is immediately followed by the substitution array. For each substitution, it lists size and data type (see Fig. 3), and uses actual value to fill into the corresponding substitution token to comprise complete plain text XML. For one chunk, most of records only

[2] Record ID is the same as record number.

have a reference of the template to reduce storage space. Probably a record in the fragment cannot be recovered for its dependence of the template. It is observed that the count of the substitution array is 18 or 20. Additionally, length and type should be followed by the hexadecimal value 0x00 [5]. Therefore we can locate the position of the substitution array, and even determine whether the record is complete by checking integrity of the substitution array.

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
        <System>
        ...
        <EventID>substition 3, type 6</EventID>
        ...
        <EventRecordID>substition 10, type 10<EventRecordID>
        </System>
</Event>
```

Fig. 2. Template with unfilled substitution array

The number of data(18 or 20)		
Length[0]	Type[0]	0x00
... ...		
Length[n-1]	Type[n-1]	0x00
Data[0]		
... ...		
Data[n-1]		

Fig. 3. Substitution array

4 The Proposed Approach

In this section, it is proposed to introduce our algorithm showed in Fig. 4. The algorithm mainly includes three parts: pre-processing data, reassembling fragments and extracting corrupted records.

4.1 Data Pre-processing

In this stage, the fragments belonging to logs should be effectively classified with others. The fragmentation points which normally bring challenge in file carving can only be present at the boundary between two clusters [2]. Since the log data may be scattered in any part of the image, we need to locate all the fragments belonging to EVTX logs by using different *magic string* to finding the first cluster of the fragment. We recommend to use 4 KB cluster as the size of per scanning, since 4 KB cluster is default for all NTFS file systems since Windows NT 4.0.

Separate lists are designed base on the mentioned file structures. Each included element of the lists which can be regraded as the fragment will store corresponding binary data. Figure 5 illustrates the flowchart of data pre-processing, and these lists are as follows:

- File list (simply as $List_f$): in $List_f$, each $element_f$ as the start of file contains a file header, a chunk header and included records.
- Chunk list (simply as $List_c$): in $List_c$, each $element_c$ as an potential chunk contains a chunk header and included records.
- Record list (simply as $List_r$): in $List_r$, each $element_r$ is regarded as an assemblage of fragmented records.

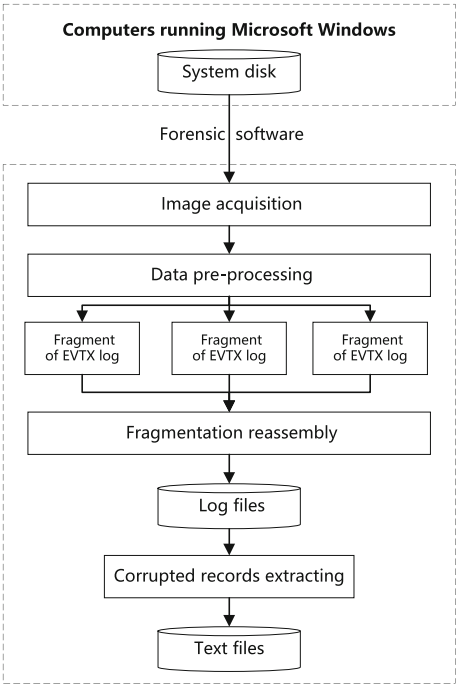


Fig. 4. Illustration of architecture

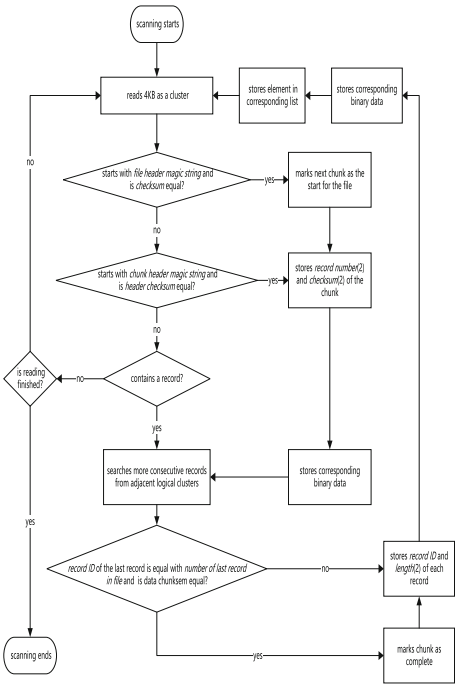


Fig. 5. The flowchart of data pre-processing

4.2 Fragmentation Reassembly

Before reassembly, we need to process the pinpointed fragments and reassemble them to reconstruct original files. Only the complete chunks can be combined into a valid log file, so we have to recover chunks belonging to the original file in the first step.

Afterwards, we generate log files by using complete chunks. Field *Channel* of the binary XML is used to determine whether a chunk belongs to the original file. It should be noted that the value of field *Channel* is not only stored in the substitution array but also in template. In order to acquire templates, we need to adopt XML technology to parse the complete chunks based on the previous research [5,6]. If one element cannot be used to reassemble finally, it will be added to $Broken\,list$.

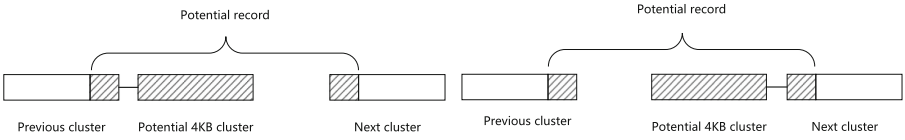
For clarity, we introduce a discriminator for merging and a simplified algorithm is presented in Algorithm 1.

– Record ID: the *record ID* sequence of records in one chunk will be consistent and two adjacent chunks are supposed to have consecutive *record ID*.

– Channel: probably two logs have many same *record ID*, but different chunks from the same file will have the same value of the field *Channel* which can be used to reassemble chunks from the same log.
– Integrity of the substitution array: if *length* of the record which is to be connected is larger than 4 KB, the only way is to try all the situations of fragmentation to check the integrity of the substitution array. A simple instance uses Fig. 6 to illustrate it. If the size of uncertain data is 4 KB, we need to determine which cluster the potential 4 KB cluster is adjacent to the previous cluster or the next cluster by verifying the integrity of the substitution array.
– Checksum: we need to calculate the checksum of all the records data belonging to this chunk when finding the last record of the chunk.

Algorithm 1. Fragmentation Reassembly Algorithm

```
Input: List_f, List_c, List_r
Output: Log files, Broken list
 for element_f, element_c ∈ List_f, List_c do
    for element_r ∈ List_r do
       merge element_r into element_f, elment_r based on discriminator
       if the last record of the chunk is found then
          mark element_f, lement_c as complete
       end if
    end for
 end for
 parse templates of the complete chunks
 for element_f ∈ Complete list_f do
    for element_c ∈ Complete list_c do
       merge element_c into list_f based on discriminator
       generate a log file using corresponding binary data
    end for
 end for
 Broken list ← rest of elment
 return Log files, Broken list
```



(a) The potential 4KB cluster is adjacent to the previous cluster

(b) The potential 4KB cluster is adjacent to the next cluster

Fig. 6. Reassembly of a record larger than 4 KB

4.3 Corrupted Records Extracting

Since EVTX log have three types of checksum to verify the integrity of a EVTX format file, any corruption results in that a log cannot be open by Windows. And a corrupted log file make its fragments not be merged. The only way to collect information of corrupted files is to match original templates and store

generated plain text XML in other format files (e.g. text file). A warning is that this process may recover the incorrect records which are generated by Windows event logging service randomly.

Experimentally, the same template shared by different records in the same chunk must have only one *template id*. Only if the type of each substitution is compatible with template can the substitution array use value to fill into the corresponding substitution tokens. For each record in the $Broken\,list$, we consider a brute force approach to search its original template and write plain text XML into a text file.

5 Experiment and Evaluation

These experiments are designed to demonstrate the effectiveness of caving algorithm in dealing with the situation of unavailable file system metadata. In Windows, all computers event logs are normally found in: $C:Windows\backslash System32\backslash winevt\backslash Logs\backslash$. Due to the limitation of the public Windows images, we use our own 20 GB system disk images collected from three operating systems (Windows 7, Windows 8 and Windows 10). Note that, we use WinHex[3] to acquire the system disk image of computers for guaranteeing the reliability and integrity of raw data [14].

First of all, we save original files for calculating accuracy. We use regular deletion method to remove all the log files and make forensic images of system disk from each operation systems. The common evaluation method is to compare whether there exists the same record. First, all the records acquired from the original log files are to be gathered manually and analysed statistically. Then we use the same method in the recovered log files. Finally, by comparing the records from the original log files with recovered ones, we can determine whether the experimental result is effective or not.

We draw support from EnCase which is a widely-used commercial forensic software utilized by some law enforcement agencies. Unfortunately, no records are recovered by EnCase for the dependance of system metadata. Because of three types of verification in the EVTX format file, the recovered log files can guarantee their correctness. Zero-error carving strategy means that we only try to recover complete log files as far as possible, thus no error records will be recovered. Complete carving strategy means we also recover records from the log files which are overwritten or corrupted during deleting to write plain text XML into text files. Moreover, the precision rate might decrease with the increasement of recall rate. All things considered (see Table 1)[4], if there can exist error records, we recommend to use complete carving during investigation for better comprehensive evaluation.

[3] WinHex is a disk editor and a hex editor useful in data recovery and forensics.

[4] We use R/O(Recovered/original), PR(Precision rate), RR(Recall rate), F(F-value) and Time to evaluate the quality of results accurately.

Table 1. Results of different carving strategies

(a) After zero-error carving

System	R/O	PR	RR	F	Time
Win 10	15105/15248	100%	99.06%	99.53%	158s
Win 8	5124/6020	100%	85.11%	91.96%	159s
Win 7	4006/5210	100%	76.89%	86.94%	162s

(b) After complete carving

System	R/O	PR	RR	F	Time
Win 10	15292/15248	98.85%	99.13%	98.99%	160s
Win 8	5777/6020	100%	95.96%	97.94%	165s
Win 7	4842/5210	97.44%	90.57%	93.88%	171s

6 Summary

Since EVTX log files have tremendous forensic potential data in Windows forensic investigation, we present a carving algorithm for them without using the file system metadata in this paper. The traditional recovery method is highly dependent on file system metadata, thus the deleted files can not be recovered. By exploring the characteristics of Windows XML event log files, we design a caving algorithm to recover fragmented log files and extract corrupted records into text files. The numerical experiments reveal that our algorithm can perform well under the situation that log files are fragmented even corrupted.

Acknowledgment. This work is supported by the National Key R&D Plan of China under grant no. 2016YFB0800201, the Natural Science Foundation of China under grant no. 61070212 and 61572165, the State Key Program of Zhejiang Province Natural Science Foundation of China under grant no. LZ15F020003, the Key research and development plan project of Zhejiang Province under grant no. 2017C01065, the Key Lab of Information Network Security, Ministry of Public Security, under grant no. C16603.

References

1. Sharma, H., Sabharwal, N.: Investigating the implications of virtual forensics. In: 2012 International Conference on Advances in Engineering, Science and Management (ICAESM), pp. 617–620. IEEE (2012)
2. Garfinkel, S.L.: Carving contiguous and fragmented files with fast object validation. Digit. Invest. **4**, 2–12 (2007)
3. Murphey, R.: Automated windows event log forensics. Digit. Invest. **4**, 92–100 (2007)
4. Al-Nemrat, A., Ibrahim, N., Jahankhan, H.: Sufficiency of windows event log as evidence in digital forensics. University of East London, London
5. Schuster, A.: Introducing the Microsoft Vista event log file format. Digit. Invest. **4**, 65–72 (2007)
6. Xiaoyu, H., Shunxiang, W.: Vista event log file parsing based on XML technology. In: 4th International Conference on Computer Science & Education, ICCSE 2009, pp. 1186–1190. IEEE (2009)
7. Talebi, J., Dehghantanha, A., Mahmoud, R.: Introducing and analysis of the Windows 8 event log for forensic purposes. In: Garain, U., Shafait, F. (eds.) IWCF 2012/2014. LNCS, vol. 8915, pp. 145–162. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20125-2_13

8. Do, Q., Martini, B., Looi, J., Wang, Y., Choo, K.-K.: Windows event forensic process. In: Peterson, G., Shenoi, S. (eds.) DigitalForensics 2014. IAICT, vol. 433, pp. 87–100. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44952-3_7
9. Mikus, N.: An analysis of disc carving techniques. Technical report, DTIC Document (2005)
10. Richard III, G.G., Roussev, V.: Scalpel: a frugal, high performance file carver. In: Refereed Proceedings of the Digital Forensic Research Workshop, DFRWS 2005, pp. 1–10, Astor Crowne Plaza, New Orleans, Louisiana, USA, August (2005)
11. Karresand, M., Shahmehri, N.: Reassembly of fragmented JPEG images containing restart markers. In: European Conference on Computer Network Defense, EC2ND 2008, pp. 25–32. IEEE (2008)
12. Na, G.-H., Shim, K.-S., Moon, K.-W., Kong, S.G., Kim, E.-S., Lee, J.: Frame-based recovery of corrupted video files using video codec specifications. IEEE Trans. Image Process. **23**(2), 517–526 (2014)
13. Cohen, M.I.: Advanced carving techniques. Digital Invest. **4**(3), 119–128 (2007)
14. Boddington, R., Hobbs, V., Mann, G.: Validating digital evidence for legal argument, p. 42 (2008)