Recovery method of deleted records and tables from ESE database

Jeonghyeon Kim, Aran Park, Sangjin Lee

Center for Information Security Technologies (CIST), Korea University, Anam-Dong, Seongbuk-Gu, Seoul, Republic of Korea

Abstract

The Extensible Storage Engine (ESE) database is a data storage technology developed by Microsoft. It is mainly used by Windows OS and its web browser. It is possible to easily delete a table or a record in the database using the ESENT API. However, there are insufficient papers and relevant information how about recovering deleted records. Previous works apply only to some tables and fail to recover deleted data perfectly. In this paper, we analyzed the structure of the ESE database and present a general-use technique to recover deleted records and tables. We developed a tool to implement the technique, and assessed the performance of the proposed tool.

Introduction

The Extensible Storage Engine (ESE) database has been used mainly in web browsers (e.g. Internet Explorer, and Edge) and Window systems (e.g. Windows Search, and System Resource Usage Monitor). In terms of forensics, research on ESE database is important because the database is used to save and manage the main records of systems and users in the Window OS.

For example, Windows Search uses the ESE database to maintain files, emails, programs and Internet history (Chivers and Hargreaves, 2011). Also Microsoft Edge and Internet Explorer version 10 and 11 use it to save the web browsing history and information about temporary files (Chivers, 2014; Gratchoff and Kroon, 2015). Numerous other Windows components such as Windows Mail, Active Directory, Windows Live and Windows Update use it.

As with other databases, it is possible to delete records in the database in order to maintain the latest data and to efficiently manage storage space. Tables or records can be easily deleted from the database using the ESENT API (Microsoft ESENT API; Kim). The ability to obtain information about deleted records often more important than information about normal records in the database. This paper explores a universal recovery method for deleted ESE database records and presents the experimental results through development of tool.

This paper is organized as follows. Chapter 2 discusses related work on ESE database structure analysis and deleted record recovery. Chapter 3 details the database structure. Chapter 4 examines the internal changes after deleting records. Chapter 5 presents a technique to recover deleted records based on information observed in Chapter 4. Chapter 6 discusses our tool that implements the technique, and assesses its performance. Finally, Chapter 7 concludes this paper and introduces future work.

Related works

Metz analyzed the schema of many ESE databases which is used in Window Search, Windows Help and Support Services, Windows Mail, Windows Search, Windows Security, and Windows Update files (Metz). Metz analyzed the file format of the ESE database, but some
areas were only partly explained. Thus, it is difficult to fully grasp the database based on his work alone (Metz).

To recover deleted records in database, the study of the ESE database format is essential and additional analysis is required.

There has been research on ESE database file recovery applied to Windows Search and Internet Explorer. However, this method applies only to some tables and cannot recover the last column entry (Chivers and Hargreaves, 2011; Chivers, 2014).

Gratchoff and Kroon studied a built-in browser called Edge, since Windows version 10, and found that there was a great deal of similarity in where and how artifacts are saved, with the previous version of Internet Explorer (Gratchoff and Kroon, 2015). This indicates that the ESE database is used in the latest version of Windows OS.

Our previous work in this area also had some issues (Kim et al., 2015). Without performing experimentation on the recovery of deleted records and tables in an ESE database, we only recovered unused records. The prior analysis of long value page record format was not accurate, resulting in errors during recovery. We were also previously unable to recover deleted tables.

**ESE database format analysis**

To classify the deleted area in the database file and to recover records in that area, the structure of the record should be exactly analyzed. So, we studied the ESE database format based on Metz’s works. In this section, we describe the structure that should be known for recovering deleted records or the part that is not in Metz’s works.

ESE database files are comprised of multiple pages except for the database header, and Pages are managed in a B-Tree structure (Microsoft). Fig. 1 illustrates internal structure of the ESE database.

**Table**

There are multiple tables inside the ESE database, and table information is managed by a catalog table called MSysObject (Metz). Certain tables include sub-tables called LV, in order to save big-sized data. Tables have their own identification numbers. This is specified in all page headers in tables. The sub-tables called LV also use distinct identification numbers, and are managed in the same way as the superordinate table.

**Page**

A page is a logical unit used to save and manage records in the ESE database, and composed of header, data, and tag areas. A tag exists at the end of a page and increases in a reverse order. This value has the offset and the size of records, and 2 bytes are allotted for each.

Several page types have been identified, including root, data, branch, empty, space tree, index, long value; pages can be distinguished by page flag values. A record storage method is page-type dependent, and can be identified by analyzing data, branch, and long value pages.
Fig. 2. Record format in data page.

Fig. 3. Record format in branch page.
Long value page

A specific type of table for large data uses sub-tables by using pointers, without saving big data directly in records. A Long Value (LV) table uses LV pages, and saves data by using multiple records when the size of data is bigger than that of a page.

LV pages record data using a LV header and multi separated data. Fig. 4 shows the structure of a record in an LV page.

Verifying changes after deleting records

To verify changes after deleting record, we developed a tool (Kim). We then deleted records using a ‘Microsoft.Isam.Esent.Interop.Api’ class function called ‘Jet-Delete’. As a result, the B-tree structure was modified and the tag area was changed. When we deleted many records or removed a specific table using the ‘JetDeleteTable’ function, some pages were no longer needed. While the data area and the tag area were not deleted, the value changed for the available page tag item in the header area, which represents the number of records. If the pages are not used, the Empty page flag was set in the Page flags item.

Proposed record recovery technique

There are two reasons that it is possible to recover deleted records in the ESE database file. Firstly, if a data page or LV page are turned into a branch page, the pre-existing data remains in the branch page. Secondly, when records are deleted, the tags and the data are not deleted; only the number of records and the kind of pages are changed.

Fig. 5 illustrates a flow chart which expresses the recovery procedures of deleted records in the ESE database file. In the beginning of the ESE database file, the database header and its copy exist. Basic information is recorded in the database header such as the unique signature, the page size, the file state, the version, etc. The page offset and the record storage method are determined by the version and the page size.

For recovering records, the schema information is required about a page to determine what belongs to a certain page. Deleted tables and related records can exist in the data page of the catalog table. Thus, the schema information of deleted tables can be identified if the deleted records of the catalog table are recovered.

There may be some unrecorded pages in the branch page because they are not now in use by the ESE database.
The starting offset of deleted records can be found through the table schema information and the record structure. Data records are made up of the record header, the last fixed size data ID, the last variable size data ID, the first variable size data offset, the fixed size data, the variable size data, and the tagged data. The start offset of deleted records can be found by using the range of possible values for the record header, the last fixed size data ID, the last variable size data ID, and the first variable size data offset. Table 1 describes how to find the starting offset of deleted records by using those items.

Records can be recovered through already-known table information. For example, the container table of the WebCacheV01.dat file which IE10, IE11, and Edge save records with a fixed structure where a fixed 4-byte value, or 0x117f7700h exists after the record header.

### Implementation and performance

In order to verify the recovery method mentioned in the previous section, we developed a tool to extract normal records in the ESE database and recover deleted records, without using the database API. The tool development environment is Python 2.7. This tool is designed to be run from a command line interface and the output file is in the SQLite format. The program is executed as follows:

```
EDBForensic.py <Input Path> <Output Path>
```

The results are extracted in one database file, and the deleted tables are recovered as 'Carved_TableName'. Fig. 6 shows the SQLite-format result of recovering deleted records from the WebCacheV01.dat file used in the Edge, by using our tool.

In order to assess how exact the performance of our tool is, we used various kinds of ESE database files which have been studied before, as the test set. Table 2 gives related information and the data collection environment.

In order to assess the performance of extracting normal records, we compared our result with that of the analysis of the EseDbViewer tool which was developed as a database-API-based tool (woanware). The result that this tool produced was equivalent to our result in terms of the number and the contents of records.

We also compared our results to those produced by the ESECareve tool designed to be a recovery tool for deleted records. ESECareve only recovers data from

---

**Table 1**

<table>
<thead>
<tr>
<th>No.</th>
<th>Conditional statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The first byte</td>
</tr>
<tr>
<td>2</td>
<td>Record size, Offset</td>
</tr>
<tr>
<td>3</td>
<td>Jump size</td>
</tr>
<tr>
<td>4</td>
<td>Last fixed size data ID</td>
</tr>
<tr>
<td>5</td>
<td>Last variable size data ID</td>
</tr>
<tr>
<td>6</td>
<td>Last variable size data offset</td>
</tr>
<tr>
<td>7</td>
<td>Last variable size data offset</td>
</tr>
</tbody>
</table>

---
WebCacheV01.dat and Windows.edb files, and only some of the tables can be recovered (Chivers and Hargreaves, 2011; Chivers, 2014). For a better side-by-side comparison, we excluded the overlapped items and only compared results against the table records that ESECarve was able to recover, so that the number of recovered records was the same. While ESECarve could not recover the final field items of all the records our tool recovered records normally. For example, The ResponseHeaders field saves important information such as visited web sites, download paths (see Fig. 7), and cache file information in WebCacheV01.dat. Because it is the final field, ESECarve failed to recover it.

We experimented on recovering deleted records by utilizing the ESE database files which have not been researched, except for WebCacheV01.dat and Windows.edb files. As a result, we could recover deleted records of those files.

Table 2
Test set to measure performance.

<table>
<thead>
<tr>
<th>File name</th>
<th>OS version</th>
<th>Path</th>
<th>Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebCacheV01.dat</td>
<td>Windows 10</td>
<td>%LOCALAPPDATA%\Spartan\Database</td>
<td>Edge</td>
</tr>
<tr>
<td>WebCacheV01.dat</td>
<td>Windows 7</td>
<td>%LOCALAPPDATA%\Microsoft\Windows\WebCache</td>
<td>Internet Explorer</td>
</tr>
<tr>
<td>Windows.edb</td>
<td>Windows 7</td>
<td>%PROGRAMDATA%\Microsoft\Search\Data\Applications\Windows</td>
<td>Windows Search</td>
</tr>
<tr>
<td>WindowsMail.MSMessageStore</td>
<td>Windows 7</td>
<td>%USERPROFILE%\AppData\Local\Microsoft\Windows Mail\WindowsMail.MSMessageStore</td>
<td>Mail</td>
</tr>
<tr>
<td>DataStore.edb</td>
<td>Windows 7</td>
<td>%WINDIR%\SoftwareDistribution\DataStore</td>
<td>Windows Update</td>
</tr>
<tr>
<td>contacts.edb</td>
<td>Windows</td>
<td>%LOCALAPPDATA%\Microsoft\Windows Live \Contacts [accountname]\version\DBStore</td>
<td>Windows Live</td>
</tr>
<tr>
<td>WLCalendarStore.edb</td>
<td>Windows</td>
<td>%LOCALAPPDATA%\Microsoft\Windows Live \Calendars [account name]\DBStore</td>
<td>Windows Live</td>
</tr>
<tr>
<td>meta.edb</td>
<td>Windows 10</td>
<td>%LOCALAPPDATA%\Microsoft\Windows \SettingSync\remotemetastore\v1</td>
<td>Windows store</td>
</tr>
<tr>
<td>meta.edb</td>
<td>Windows 10</td>
<td>%LOCALAPPDATA%\Microsoft\Windows \SettingSync\metastore</td>
<td>Windows store</td>
</tr>
<tr>
<td>CortanaCoreDb.dat</td>
<td>Windows 10</td>
<td>Packages\Microsoft.Windows.Cortana_xxxx \LocalState \ESEDatabase_CortanaCoreInstance</td>
<td>Cortana</td>
</tr>
</tbody>
</table>
Conclusion

The ESE database is used to save and manage the main records of systems and users in Windows OS. Much work has been done to recover deleted records in ESE database files; however, the previous work is limited to Internet Explorer and Windows Search, and did not solve the problems of recovering dirty-state files and final-column data. Our research builds upon prior work and knowledge to address these issues and provide a tool that is forensically meaningful.

In this paper, we analyzed the ESE database file structure to extract normal records without using the database API, and compared the results with one based on the Database API. There was no difference between the results, which means that our analysis was accurate. We also studied a recovery technique for deleted data based on the database structure, and as a result, developed a tool that can recover deleted records from various versions of files to solve the problems of the previous work. The tool is available by contacting the authors.

This tool has a limitation that it cannot recover deleted data with damaged record header which contains information about the record structure of unknown records. Deleted records without the record header can be recovered if the schema information of the table to be recovered is known. Future work will address a technique to analyze varied ESE database files used in Windows, outline table structures, and recover damaged records.

Acknowledgements

This research was supported by the Public Welfare & Safety Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2012M3A2A1051106).

References

Gratchoff James, Kroon Guido. Project Spartan forensics. Amsterdam University; 2015.
woanware, "EseDbViewer v1.0.6," http://www.woanware.co.uk/forensics/esedviewer.html.

Fig. 7. Download paths in ResponseHeaders Field.