ISSN:0975-9646

Identifying and Extracting Data from Clipboard

Ganesh N. Nadargi[1], Zakir M. Shaikh[2]

[1,2]*Nagesh Karajagi Orchid College of Engineering & Technology*
Solapur-(MH), India

Abstract— **The clipboard is a very useful tool introduced by windows and is the mechanism for transferring information from one application to another through copy and paste actions. Being able to retrieve last text, file or data copied it can be useful in many applications with invaluable information. This paper describes the windows clipboard structure and the process of retrieving copy/paste information from windows.**

Keywords— **Windows Clipboard, Copy Paste operations, Clipboard Extraction, Clipboard Content.**

I. INTRODUCTION

There are lots of applications used in day to day life which involves use of Clipboard by the users. It can be used to get the content from the application and use it for other applications. That data can be monitored if there is an application which looks for what data is copied to Clipboard, using which we can restrict users from copying unwanted data and also it can be used for memory forensics by extracting Clipboard evidence. There are lot many tools which provide to extract data from different devices but there are very less tools written to monitor or extract Clipboard data from memory. The clipboard has been part of Windows O/S family since Windows 3.1. Windows uses the clipboard to transfer information between user applications. As a result it bridges gap between user functions and O/S kernel functions. As a result finding & extracting Clipboard data requires slightly different techniques.

II. BACKGROUND

The windows clipboard is the mechanism that Microsoft windows operating system uses to allow data to be shared between applications. It first appeared in Windows 3.1, although its functionality has greatly increased since then. Table 1 shows the standard formats used by the clipboard. However, Microsoft provides ability for private data formats, formats that are application specific, and that could be registered so that other applications could transfer data in these formats.

For copying more complex data than text to the clipboard, Windows makes available several APIs which make extraction much more difficult. The original method of exchanging data between applications was dynamic data exchange (DDE). In 1990, Microsoft released object linking and embedding (OLE) enabling compound files. Compound files have most of the file in a primary format (for e.g. a Microsoft Word document) and smaller sections in one or more other formats either linked in or embedded. Microsoft then updated the (OLE) and extended to a Compound Object Model (COM) in which Uniform Data Transfer (UDT) and Drag and Drop features where added. For e.g. when a file is dragged from Windows Explorer to the Desktop, this is accomplished internally via the Windows clipboard. The functionality changed over the years, first with the creation of ActiveX and most recently with the advent of .NET framework.

TABLE I
PREDEFINED CLIPBOARD FORMATS

Constant	Value	Description
CF TEXT	0x0001	Text format. Each line ends with a CR/LF combination. Null terminated
CF BITMAP	0x0002	A handle to a bitmap
CF SYLK	0x0004	Microsoft symbolic link format
CF DIF	0x0005	Software Arts Data Interchange Format
CF TIFF	0x0006	Tagged-image file format
CF PENDATA	0x000A	Data for the pen extensions to Windows
CF RIFF	0x000B	Represents audio data more complex than can be represented in a CF_WAVE standard wave format
CF LOCALE	0x0010	The data is a handle to the Locale identifies a list of files
CF WAVE	0x000C	Represents audio data in one of the standard wave formats
CF PALETTE	0x0009	Handle to a color palette
CFx UNICODETEXT	0x000D	Unicode text format. Each line ends with a CR/LF combination. Null terminated
CF METAFILEPICT	0x0003	Handle to a metafile picture format as defined by the METAFILEPICT structure
CF OEMTEXT	0x0007	Text format containing characters in the OEM character set. Each line ends with a CR/LF combination. Null terminated
CF DIB	0x0008	A memory object containing bitmapinfo structure followed by the bitmap bits
CF ENHMETAFILE	0x000E	A handle to an enhanced meta file
CF HDROP	0x000F	A handle t_type HDROP that identifies a list of files
CF DIBVS	0x0017	A memory object containing a bitmapvsheader structure followed by the bitmap color space information and the bitmap bits

III. METHODOLOGY

There are many different ways possible to extract data from clipboard, one of which is described below using which the data can be extracted and identified. The below fig describes steps required for extracting data.

First, functions that used to access clipboard are identified. Then the clipboard data is accessed using those functions and the type of data is identified depending on its properties. The data retrieved is classified in different formats and if the required format is found in the clipboard then it is tagged. Finally the identified data is extracted and used depending on applications.

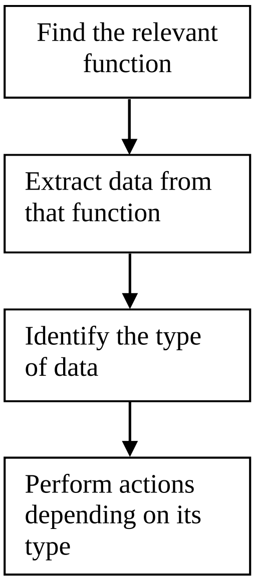
Find the relevant function
Extract data from that function
Identify the type of data
Perform actions depending on its type

Fig. 1 A process of extracting data from clipboard

A. Clipboard classes and Methods

While there is a large amount of documentation on how to use and access the Windows clipboard via application program interfaces (APIs), there is less documentation on the methods used other than APIs.

Extending classes like Clipboard we can access the data stored in the clipboard using its constructions and different methods.

The *getSystemClipboard* is called to retrieve the pointer to the clipboard data. This provides handle to the clipboard data. The data can then be used to analyse type and its properties. The *Toolkit* class is the main class which is use to get the system clipboard. A *DataFlavor* provides Meta information about data. Data Flavor is typically used to access data on the clipboard or during drag and drop operation. *FlavorListeners* may be registered on an instance of the Clipboard class to be notified about the changes to the set of *DataFlavors* available on this Clipboard. There are different data flavors that provide the type of data that is stored in the clipboard. Those flavors are classified depending on file formats.

isDataFlavorAvailable method is used to check whether the data flavor of the data in the clipboard is in the list of valid data flavors. Manipulating the clipboard data is quiet easy but it will be not useful in applications where the memory forensics is required. So care needs to be taken while handling pointer of the clipboard which is directly accessed.

B. Implementation

Using above classes and methods the data can be obtained from the clipboard. The below algorithm can be used as the basic layout for the working of extraction.

```
Clipboard c;
c=getSystemClipboard();
if(DataFlavor is Available)
getdata(DataFlavor);
```

It is possible that the user closed the application after copying data to the clipboard. Recall that the clipboard bridges user space and kernel space. While each process has a local copy of the clipboard once it has accessed the clipboard functions, the kernel also has the clipboard. Therefore, until overwritten, clipboard data for a closed process is still available in the clipboard.

It is also possible to change the contents of the clipboard using *setContents* method. But the owner on the current control is shifted to the application which modifies the content of the clipboard data. For e.g.

```
Clipboard c;
StringSelection cnt= new StringSelection(“Data”);
c.setContents(cnt);
```

As shown above the current data flavor is been replaced by the String flavor as the strings “Data” is been placed in the clipboard replacing the older data. Clipboard class is much useful for altering data in the clipboard without letting the user realise about it. It can be also used for the applications which are based on monitoring the clipboard and restricting the contents to be copied

IV. CONCLUSIONS

The above techniques used for accessing and identification of data in the clipboard are implemented successfully. As the generic classes and the methods are used it can be easily modified or enhanced with the other codes in order to use for different platforms of operating systems.

Using those accessing and retrieving methods different applications can be implemented at ease, such as clipboard managers, clipboard forensics and clipboard restriction tools.

ACKNOWLEDGMENT

This work is implemented as the part of dissertation curriculum. It is supported by N. K. Orchid college of Engineering and Technology. The views expressed in this paper are those of the authors and do not represent the views or policies of NKOCET.

REFERENCES

[1] Allan R. History of the personal computer: the people and the Technology. Allan Publishing; 200.

[2] Microsoft.com How to add data to the clipboard, http://www.microsoft.com/windowsxp/using/setup/tips/clip-book.mspx. [accessed 12.01.15].

[3] Microsoft.com OLE Background, http//msdn.microsoft.com/en-us/library/aa271002(v=VS.60).aspx.[accessed 12.01.15].
James Okolica, Gilbert L. Peterson, "Extracting the windows clipboard from physical memory", Science Direct(2011).

[4] Oracle.com Class Clipboard, http://docs.oracle.com/javase/7/docs/api/java/awt/datatransfer/Clipboard.html. .[accessed 22.03.15].

[5] Oracle.com Class Toolkit, http://docs.oracle.com/javase/7/docs/api/java/awt/Toolkit.html.[accessed 22.03.15].

[6] S. Li, S. Lv, X. Jia and Z. Shao "Application of Clipboard Monitoring Technology in Graphic and Document Information Security Protection System" published in Third International Symposium on Intelligent Information Technology and Security Informatics, IEEE 2009.