Extracting the Windows Clipboard from Physical Memory

James S. Okolica
Gilbert L. Peterson
Overview

• Cyber Forensics and Live Response
  • Why do we need it? What can we get from it?
  • The Compiled Memory Analysis Tool (CMAT)

• Windows Clipboard
  • What it does and how it works (from an API perspective)
  • Under the covers

• Integrating Clipboard into CMAT
  • User Side (user32.dll)
  • Kernel Side (win32k.sys)

• Experimental Set up and Results

• Next Steps
Live Cyber Forensics

• Business Productivity
  • Lost Revenue
  • Supervisory Control and Data Acquisition (SCADA) Systems
  • Concern of the system coming back up

• Acquisition of volatile-only information
  • Network Traffic
  • Active process and user information

• Encrypted Hard Drives

• Memory Resident Malware

• Too much data
Types of Volatile Data

- General Operating System Information
- Services/ Driver Information
- Logged on users and their authentication credentials
- Registry Information
- Process Information
  - Open Files
  - Open Registry Keys
  - Network Connections and Status
  - Dynamic Link Libraries
- Command History
- Clipboard Contents

A Compiled Memory Analysis Tool (CMAT)

- Determines O/S version (using _DBGKD_DEBUG_DATA_HEADER64 or finding the kernel PE)
  - Physical Address Extensions enabled/disabled, 32 bit/64 bit

- Loads O/S specific data structures (by retrieving PDBs from Microsoft’s Symbol Server)

- Locates O/S-agnostic signatures for processes and registries

- Connects users found in the registry with processes

- Locates data structures within PEs (by retrieving PDBs from Microsoft’s Symbol Server)
  - Network activity
  - Clipboard data
The Windows Clipboard

• Sharing data between applications
  1. Select an object and send it to a common area
  2. Retrieve the object from the common area

• Observations:
  • Only one object can be in the common area at a time
  • The object can be stored in multiple formats

• History
  • Dynamic Data Exchange (DDE)
  • Object Linking and Embedding (OLE)
  • Compound Object Model (COM)
  • Object Linking and Embedding v2.0 (UDT, Drag & Drop)
  • Active X
  • .NET
Windows Clipboard Format & Functions

- **Predefined Formats**
  - Formats identified by Microsoft when the Clipboard was initially implemented
    - CT_TEXT 0x0001
    - CF_BITMAP 0x0002
    - CF_TIFF 0x0005
    - CF_WAVE 0x000C

- **Private Formats**
  - Formats developed by vendors (including Microsoft) to enable transfer of proprietary formats (e.g., Microsoft Office objects)
    - OLE 0xC013
    - IDataObject 0xC009

- **Multi-Formats**
  - Although only one piece of data can be in the Clipboard at a time, programs can save that data in multiple formats (e.g., MS Office, OLE Object, Unicode, ASCII)

### Transferring Text to the Clipboard

```c
hGlobal = GlobalAlloc (GHND | GMEM_SHARE, iLength + 1) ;
pGlobal = GlobalLock (hGlobal);
for (i = 0; i < wLength; i++)
  *pGlobal++ = *pString++;
GlobalUnlock (hGlobal);
OpenClipboard (hwnd);
```

```c
hGlobal = GetClipboardData (CF_TEXT);
EmptyClipboard();
SetClipboardData (CF_TEXT, hGlobal);
CloseClipboard();
```

### Retrieving Text from the Clipboard

```c
OpenClipboard (hwnd);
```

```c
hGlobal = GetClipboardData (CF_TEXT);
```
Reversing Methodology

- Create a Virtual Machine (VM)
- Execute a copy/paste operation
- Perform dynamic analysis to locate the structures
- Generate a dump file of the VM’s memory and duplicate the dynamic analysis

```
Transferring Text to the Clipboard

hGlobal = GlobalAlloc (GHND | GMEM_SHARE, iLength + 1) ;
pGlobal = GlobalLock (hGlobal);

for (i = 0; i < wlength; i++)
  *pGlobal++ = *pString++;

GlobalUnlock (hGlobal);

OpenClipboard (hwnd);
EmptyClipboard();
SetClipboardData (CF_TEXT, hGlobal);
CloseClipboard();
```

```
Retrieving Text from the Clipboard

OpenClipboard (hwnd);

hGlobal = GetClipboardData (CF_TEXT);
```

GetClipboardData

Input: Format

NtUserGetClipboardData
Input: Format

CheckclipboardAccess
Output: *WindowStation

xxxGetClipboardData
Input: Format, *WindowStation
Output: Handle to Clipboard Data

CreateLocalMemHandle
Input: Handle
Output: Pointer

ntUserCreateLocalMemHandle
Input: Handle
Output: Pointer

Populate gphn data structure with clipboard data

Return pointer to gphn record with requested format

Clipboard Structure

<table>
<thead>
<tr>
<th>Offset</th>
<th>32 bit Offset</th>
<th>64 bit Offset</th>
<th>Data Type</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>gphn*</td>
<td>Next</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>0x08</td>
<td>uint16_t</td>
<td>Format</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>0x10</td>
<td>void*</td>
<td>Unknown</td>
<td></td>
</tr>
<tr>
<td>0x0c</td>
<td>0x18</td>
<td></td>
<td>Handle</td>
<td></td>
</tr>
</tbody>
</table>
User Clipboard Integration (user32.dll)

1. Find first process
2. Find first entry in loader table
   - If entry = user32.dll
     - Open user32.pdb (download if necessary)
       - Locate gphn symbol offset
       - Locate gphn in user32.dll data section
       - Check if current entry has clipboard information
         - Yes: Display clipboard information
         - No: Find next entry in gphn linked list
3. More Entries
   - Find next entry in loader table
     - No More Entries
4. Find next process (sorted by session ID)
   - More Entries
   - No More Entries
   - End
Kernel Clipboard Integration (win32k.sys)

- Iterate through PsLoadedModuleList to find win32k.sys
- Extract Debug Data & use that to retrieve win32k.pdb
- Retrieve the location of symbol gSharedInfo
- Retrieve the WindowStation pointer for the current process
  - Iterate through table of clipboard formats until the correct format is found
    - Starting Location: (32 bit: WindowStation + 0x58, 64 bit: WindowStation + 0x2c)
    - Increment: (32 bit: 0x10, 64 bit: 0x18)
    - Handle: (32 bit: offset 0x04, 64 bit: offset = 0x08)
  - Convert Handle to pointer to clipboard data
    - low16 = low 16 bits of the handle
    - 32 bit:
      - recsize = gSharedInfo + 0x08
      - baseaddress = gSharedInfo + 0x04
    - 64 bit:
      - recsize = gSharedInfo + 0x10
      - baseaddress = gSharedInfo + 0x08
    - Vista and XP 64 bit: offset = low16 * 0x03 * 0x08
    - Vista and XP 32 bit: offset = low16 * 0x03 * 0x04
    - Windows 7 64 bit: offset = low16 * recsize
    - Windows 7 32 bit: offset = low16 * recsize
    - POINTER = baseaddress + offset
- Retrieve the unicode string
  - 64 bit: POINTER + 0x14
  - 32 bit: POINTER + 0x0c
Experimental Setup

- **DFRWS 2008 Forensic Rodeo**
  - 2 Windows XP 32 bit memory dumps

- **NIST CFReDS dataset –**
  - 1 Windows Vista 32 bit memory dump
  - 2 Windows XP memory 32 bit dumps

- **Additional memory dumps**
  - 6 operating system configurations
    - Windows XP SP3 32 bit, Windows Vista (pre-SP1) 32 bit, Windows 7 SP3 32 bit
    - Windows XP SP2 64 bit, Windows Vista (pre-SP1) 64 bit, Windows 7 SP3 64 bit
  - For each operating system configuration
    - Memory dump with clipboard data from MS Excel 2007
    - Memory dump with clipboard data from MS Word 2007
    - Memory dump with clipboard data from Notepad
## Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Memory Image</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFRWS2008</td>
<td>Dfrws</td>
<td>“Pp –B –p –o out.pl file” command found</td>
</tr>
<tr>
<td>CFReDS</td>
<td>Vista-beta2.img</td>
<td>No Clipboard Data Found</td>
</tr>
<tr>
<td>CFReDS</td>
<td>Xp-laptop-2005-06-25.img</td>
<td>No Clipboard Data Found</td>
</tr>
<tr>
<td>CFReDS</td>
<td>Xp-laptop-2005-07-04-1430.img</td>
<td>Non-textual Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit XP w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit XP w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit XP w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit XP w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit XP w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit XP w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
</tbody>
</table>
# Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Memory Image</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generated</td>
<td>32 bit Vista w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit Vista w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit Vista w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Vista w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Vista w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Vista w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit Vista w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit Win7 w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>32 bit Win7 w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Win7 w/ Notepad</td>
<td>Notepad Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Win7 w/ MS Word</td>
<td>MS Word Clipboard Data Found</td>
</tr>
<tr>
<td>Generated</td>
<td>64 bit Win7 w/ MS Excel</td>
<td>MS Excel Clipboard Data Found</td>
</tr>
</tbody>
</table>
Next Steps

• Proprietary/ Application Specific Formats
  • IDataObjects
  • OLE Objects

• Different ways to copy data
  • Between applications
  • Within an application
  • Drag and Drop

• Formalizing a process for reversing DLLs and Drivers
Questions