



Extracting Windows Command Line Details from Physical Memory

By

Richard Stevens and Eoghan Casey

Presented At

The Digital Forensic Research Conference

DFRWS 2010 USA Portland, OR (Aug 2nd - 4th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

<http://dfrws.org>

*Extracting Windows Command
Line Details from Physical
Memory.*

Richard Stevens and Eoghan Casey

Introduction

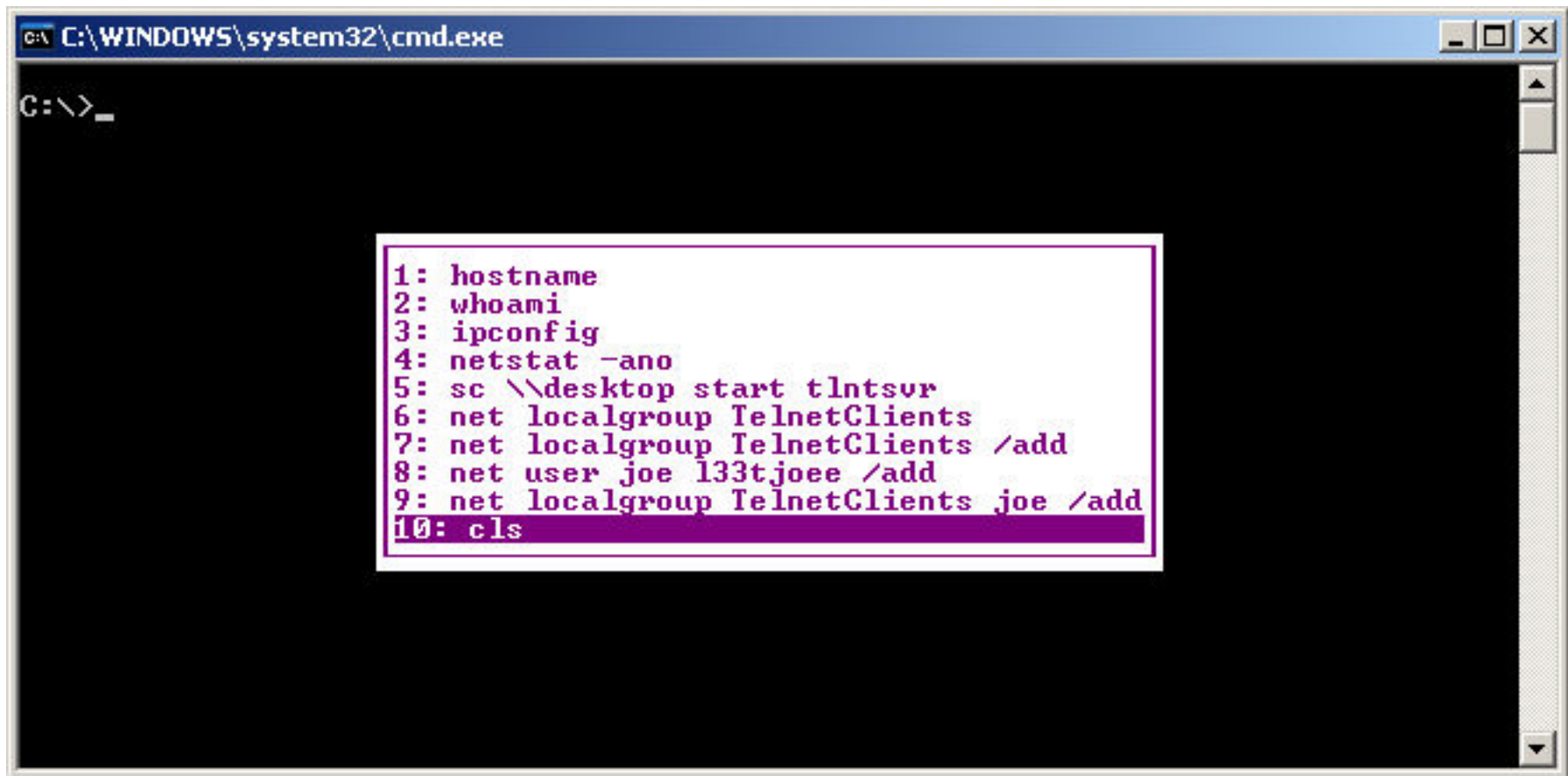
- Recent research has demonstrated how effective memory forensics can be in identifying information of forensic value.
- One area of interest is the commands that have been typed by a user or attacker into the command prompt.
- The presence of a command string in memory *might* indicate a command that was typed by the user. Or it could be a fragment of a help file.
- Context is important - even if commands can be identified using simple string searches the order can change the meaning
- Is it possible to recover the history of a command prompt from a Windows XP memory capture by examining the data structures used to store the command history?

Windows Command Line History

- DOSKEY

- Originally a separate application for MS-DOS that has been incorporated into Windows XP as a command.
- Accessed by entering “doskey / history” or pressing “F7”
- stores the last 50 commands entered by the user by default
- The buffer’s size is stored in:
HKEY_CURRENT_USER\Console\HistoryBufferSize
- Most familiar to DOS and Windows users using the “up” and “down” arrows to scroll through previous commands.
- Memory resident and only normally accessible from within a running cmd.exe process.
- Can be cleared using “doskey /reinstall” or pressing “ALT-F7”

DOSKEY



The image shows a Windows command prompt window with a black background and white text. The title bar at the top reads "C:\WINDOWS\system32\cmd.exe". The prompt "C:\>_" is visible in the top left. A white rectangular box with a purple border is centered on the screen, containing a list of ten DOSKEY macros, each numbered from 1 to 10. The macros are: 1: hostname, 2: whoami, 3: ipconfig, 4: netstat -ano, 5: sc \\desktop start tlntsvr, 6: net localgroup TelnetClients, 7: net localgroup TelnetClients /add, 8: net user joe l33tjoe /add, 9: net localgroup TelnetClients joe /add, and 10: cls. The 10th macro, "10: cls", is highlighted with a purple background.

```
C:\WINDOWS\system32\cmd.exe
C:\>_

1: hostname
2: whoami
3: ipconfig
4: netstat -ano
5: sc \\desktop start tlntsvr
6: net localgroup TelnetClients
7: net localgroup TelnetClients /add
8: net user joe l33tjoe /add
9: net localgroup TelnetClients joe /add
10: cls
```

Initial Research

- A literary review was conducted to identify any indications of how DOSKEY structures might be stored. No significant resources were found.
- A Windows XP VM was created and memory captures were taken using 256mb, 512mb, and 1024mb of RAM.
 - Each capture contained one or more command prompt windows containing known commands including readily identifiable strings.
 - The resulting image was examined manually to identify possible data structures containing the known unique commands.
- Once possible command history elements were identified in memory, each was tested and verified by poking the memory of a live VM and observing the results on the command history.

Initial Findings

- Even knowing a unique command string, still difficult to identify the underlying data structure
- Data structures are not stored within the cmd.exe process as expected
 - In Windows XP DOSKEY the data structures containing the command history are found within the Windows XP user runtime process (*csrss.exe*)
- *Commands* are stored in a relatively simple data structure encoding the length of the command and a Unicode representation of the command.
- The *Command History* is maintained in a data structure that encodes the number of command elements and pointers to each individual command.

Command History

commandElement {

<i>0x00</i>	<i>short</i>	<i>ByteCount;</i>	<i>// Short, Little-Endian</i>
<i>0x02</i>	<i>char</i>	<i>Command [ByteCount/2];</i>	<i>// UTF-16</i>

}

commandHistory {

<i>0x00</i>	<i>short</i>	<i>ElementCount;</i>
<i>0x02</i>	<i>short</i>	<i>endOffset;</i>
<i>0x04</i>	<i>short</i>	<i>pointerIndex;</i>
<i>0x06</i>	<i>short</i>	<i>startOffset;</i>
<i>0x08</i>	<i>short</i>	<i>HistoryBufferSize;</i>
<i>0x16</i>	<i>commandHistory*</i>	<i>?</i>
<i>0x20</i>	<i>commandHistory*</i>	<i>?</i>
<i>0x24</i>	<i>commandElement1*</i>	
<i>0x28</i>	<i>commandElement2*</i>	
<i>0x n</i>	<i>commandElementHistoryBufferSize*</i>	

}

Evaluation of Method

- Unable to identify the function of all fields within the *Command History* structure.
- Each *Command Element* is useful from a forensic perspective, but not unique enough to identify in a memory image with any significant success.
- Possible to search for commands using a *Bottom-up* approach by identifying possible command elements, then searching for the corresponding *Command History* object.
- Unfortunately this required a time intensive manual reconstruction process that was difficult to scale.
- Each *Command History* structure is fairly unique.
- If *HistoryBufferSize* is known and intact it is relatively easy to scan for these structures using a *top-down* approach.

Top-down Method

- Scan for Command History Objects, then enumerate the array of pointers to each Command History Element.

commandHistory {

0x00short Between 0 and HistoryBufferSize
0x02short Between -1 and HistoryBufferSize
0x04short Between -1 and HistoryBufferSize
0x06short Between 0 and HistoryBufferSize
0x08short HistoryBufferSize (default 0x32);
0x16commandHistory Valid Address*
0x20commandHistory Valid Address*

}

Evaluation of Method

- A volatility plug-in was written called “*cmd_history*” that scans for and displays possible command history objects using a *top-down* approach
- The plug-in was then tested against three publicly available Windows XP memory captures with an unknown command history.
 - DFRWS 2008 Rodeo Image
 - *dfrws2008-rodeo-memory.img*
 - NIST Reference Data Set
 - *xp-laptop-2005-07-04.img*
 - *xp-laptop-2005-06-25.img*
- In all three samples we were able to recover an intact command history object that contains large segments of commands.

DFRWS 2008 Rodeo

Element Count	End Offset	pointer Index	Start Offset	History Buffer Size
4	3	3	0	50

Virtual Address	Physical Address	Size (bytes)	Command
004E8E88	149cfE88	8	cd \
01283A20	14fbbA20	18	mkdir mem
01283B48	14fbbB48	12	cd mem
004E1FF8	80d6FF8	138	"Z:\emidnight On My Mac\Downloads \mdd_1.3.exe" -o sv-laptop-memo
<i>Add. Command</i>			
012839C0	14fbb9C0	88	(vxfer.exe X:\Secretplans\secretplans1.jpg
01283AE8	14fbbAE8	-exe X:\Secretplans\secretplans.....
01283B48	14fbbB48	12	cd mem
01283BA8	14fbbba8	84	svxfer.exe X:\Secretplans\secretplans7.jpg
004E1FA0	80d6FF8	?	?

C:\WINDOWS\system32\cmd.exe

Possible Command History Buffer - Physical Address: 0x152f9d98

Size	Elements	Start	End	Pointer
50	4	0	3	3

Buf	Virtual	Physical	Size	Command
0	0x4e8e88	0x149cfe88	8	cd \XE Jw? NNO , 8 i ä G N
1	0x1283a20	0x14fbb20	18	mkdir mem
2	0x1283b48	0x14fbbb48	12	cd mem. Sve N Nsecrctplans\secretplans5.jp
3	0x4e1ff8	0x80d6ff8	138	"Z:\emidnight On My Mac\Downloads\mdd_1.3
4	0x12839c0	0x14fbb9c0	88	suxfer.exe X:\Secretplans\secretplans1.jp
5	0x1283ae8	0x14fbbae8	0	.exe X:\Secretplans\secretplans 0p
6	0x1283b48	0x14fbbb48	12	cd mem. Sve N Nsecrctplans\secretplans5.jp
7	0x1283ba8	0x14fbbba8	84	suxfer.exe X:\Secretplans\secretplans7.jp
8	0x4e1fa0	0x80d6fa0	0	\WINDOWS\system
9	0x12834c0	0x14fbb4c0	99	\WINDOWS\system32\cmd.exe - cd mem em !!
10	0x0	0x7fe0000	0	md.exe
11	0x0	0x7fe0000	0	
12	0x0	0x7fe0000	0	
13	0x0	0x7fe0000	0	

NIST – XP Laptop 2005/06/25

Element Count	End Offset	pointer Index	Start Offset	History Buffer Size
7	6	6	0	50

Virtual Address	Physical Address	Size (bytes)	Command
004E2D28	14400D28	4	d:
004E1F78	dcbfF78	12	cd dd
004E2CC8	14400CC8	6	dir
004E2E00	14400E00	34	Cd UnicodeRelease
004E2CB8	14400CB8	6	dir /
004E1F90	dcbfF90	6	dd
004E1FF8	dcbfFF8	88	dd (presumably dd memory image command)

NIST – XP Laptop 2005/07/04

Element Count	End Offset	pointer Index	Start Offset	History Buffer Size
20	13	13	0	50

Virtual Address	Physical Address	Size (bytes)	Command
004E1F90	de7fF90	4	dd
004E2CB8	193ecCB8	6	cd\
004E2D18	193ecD18	4	dr
004E2D28	193ecD28	6	ee:
004E2D38	193ecD38	4	e;
004E2D48	193ecD48	4	e:
004E2D58	193ecD58	4	dr
004E2D68	193ecD68	4	d;
004E2D78	193ecD78	4	d:
004E2D88	193ecD88	4	dr
004E2D98	193ecD98	4	ls
004E2Da8	193ecDa8	14	cd Docu
004E2DC0	193ecDC0	68	cd Documents and.....
004E2E58	193ecE58	4	dr
004E2E68	193ecE68	4	d:
004E2E78	193ecE78	12	cd dd\
004E2E90	193ecE90	34	cd UnicodeRelease
004E2Ec0	193ecEc0	4	dr
004E2ED0	193ecED0	6	dd
004E4100	19588100	132	dd if=\\.\PhysicalMemory of=c:\xp-2005-07-04-1430.img conv=noerror

Interesting Results

- The command history “size” field may allow the contents of a fragmented or partial command string to be inferred by allowing us to identify the size of the original command.

Virtual Address	Physical Address	Size (bytes)	Command
004E1FF8	dcbfFF8	88	dd (presumably dd memory image command)

xp-laptop-2005-06-25.img

Virtual Address	Physical Address	Size (bytes)	Command
004E2DC0	193ecDC	68	cd Documents and.....

xp-laptop-2005-07-04.img

Interesting Results

- The command history buffer may contain pointers to commands from a wiped or closed command prompt session (slack space).

Virtual Address	Physical Address	Size	Command
004E8E88	149cfE88	8	cd \
01283A20	14fbbA20	18	mkdir mem
01283B48	14fbbB48	12	cd mem
004E1FF8	80d6FF8	138	"Z:\emidnight On My Mac \Downloads \mdd_1.3.exe" -o sv- laptop-memo
012839C0	14fbb9C0	88	(vxfer.exe X:\Secretplans \secretplans1.jpg
01283AE8	14fbbAE8	-exe X:\Secretplans \secretplans.....
01283B48	14fbbB48	12	cd mem
01283BA8	14fbbba8	84	svxfer.exe X:\Secretplans \secretplans7.jpg
004E1FA0	80d6FF8	?	?

Element Count
4

Interesting Results

- In some cases, we were able to recover the command history objects from a closed *cmd.exe* processes.

```
C:\WINDOWS\system32\cmd.exe

C:\Volatility>python volatility pslist -f Bob.vmem
C:\Volatility\forensics\win32\crashdump.py:31: DeprecationWarning: the sha module
is deprecated; use the hashlib module instead
  import sha
C:\Documents and Settings\tssah50\AppData\Python\Python26\site-packages
\Crypto\Hash\MD5.py:6: DeprecationWarning: the md5 module is deprecated; use has
hlib instead
  from md5 import *
Name      Pid  PPid  Thds  Hnds  Time
System    4    0     58   573   Thu Jan 01 00:00:00 1970
smss.exe  548  4      3    21   Fri Feb 26 03:34:02 2010
csrss.exe 612  548   12   423   Fri Feb 26 03:34:04 2010
winlogon.exe 644  548   21   521   Fri Feb 26 03:34:04 2010
services.exe 688  644   16   293   Fri Feb 26 03:34:05 2010
lsass.exe 700  644   22   416   Fri Feb 26 03:34:06 2010
vmacthlp.exe 852  688    1    35   Fri Feb 26 03:34:06 2010
svchost.exe 880  688   28   340   Fri Feb 26 03:34:07 2010
svchost.exe 948  688   10   276   Fri Feb 26 03:34:07 2010
svchost.exe 1040 688   83   1515  Fri Feb 26 03:34:07 2010
svchost.exe 1100 688    6    96   Fri Feb 26 03:34:07 2010
svchost.exe 1244 688   19   239   Fri Feb 26 03:34:08 2010
spoolsv.exe 1460 688   11   129   Fri Feb 26 03:34:10 2010
vntoolsd.exe 1628 688    5   220   Fri Feb 26 03:34:25 2010
VMUpgradeHelper 1836 688    4   108   Fri Feb 26 03:34:34 2010
alg.exe 2024 688    7   130   Fri Feb 26 03:34:35 2010
explorer.exe 1756 1660  14   345   Fri Feb 26 03:34:38 2010
VMwareTray.exe 1108 1756    1    59   Fri Feb 26 03:34:39 2010
VMwareUser.exe 1116 1756    4   179   Fri Feb 26 03:34:39 2010
uscntfy.exe 1132 1040    1    38   Fri Feb 26 03:34:40 2010
msiexec.exe 244  688    5   181   Fri Feb 26 03:46:06 2010
msiexec.exe 452  244    0   -1    Fri Feb 26 03:46:07 2010
vmauc1t.exe 440  1040    8   188   Sat Feb 27 19:48:49 2010
vmauc1t.exe 232  1040    4   136   Sat Feb 27 19:49:11 2010
firefox.exe 888  1756    9   172   Sat Feb 27 20:11:53 2010
AcroRd32.exe 1752 888    8   184   Sat Feb 27 20:12:23 2010
svchost.exe 1384 688    9   101   Sat Feb 27 20:12:36 2010

C:\Volatility>
```

Possible Command History Buffer - Physical Address: 0x1c506d98

Size	Elements	Start	End	Pointer
50	2	0	1	1

Buf	Virtual	Physical	Size	Command
0	0x12632d8	0x1c2c82d8	216	pconfig
1	0x4e1eb8	0x54b6eb8	40	ing google.com!!

bob.vmem

https://www.honeynet.org/challenges/2010_3_banking_troubles

Interesting Behaviors

- Separate command history buffer objects are created for each *cmd.exe* process on the system.
- DOSKEY /reinstall effective at removing “live” DOSKEY history structures.
- In practice however partially intact copies may be found in memory even after deletion.
- Default and “erased” command history buffer objects are identical.

Challenges & Limitations

- In practice can be difficult to capture closed command history prompts in an actual intrusion or forensic investigation unless the cmd.exe process is still active.
- The current search relies on knowledge of the maximum buffer size which can be changed by the end user.
- Research is limited to Windows XP Machines. Ongoing work indicates that there are identical structures are present in Windows 2000 and Windows Server 2003.
- A properly written bottom-up approach may be more effective at identifying partially overwritten command history structures.
 - Frequency analysis of DOS Commands?
- Several variables within the command history structure are unidentified and if altered may alter the effectiveness of the process.

Conclusion

- Relatively easy to scan for intact DOSKEY command data structures once the structures themselves have been identified using a top-down approach.
- Information of forensic value can be recovered from both complete and partially intact DOSKEY structures.
- DOSKEY structures exhibit slack space that may contain information of interest.
- DOSKEY metadata can be used to infer the number, length, content, or order of partially recovered commands.
- Work needs to be expanded to identify similar structures in Windows Vista, Windows 7, Server 2008 and alternate command shells such as *Windows PowerShell*.

Questions?