Searching for Processes and Threads in Microsoft Windows Memory Dumps

By
Andreas Schuster

Presented At
The Digital Forensic Research Conference
DFRWS 2006 USA Lafayette, IN (Aug 14th - 16th)

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
Searching for Processes and Threads in Microsoft Windows Memory Dumps.

Andreas Schuster
Deutsche Telekom AG
Group Security
andreas.schuster@telekom.de
Searching for Processes and Threads.

Agenda.

1. Introduction

2. Searching for Objects
   2.1 Memory Allocations
   2.2 Kernel Object
   2.3 EPROCESS / ETHREAD

3. Proof of Concept – PTfinder

4. Conclusion

5. Questions & Answers
Introduction.
Development of Memory Forensics in 2005.

Why memory forensics?

- certain attacks don‘t leave traces on disk
- Which processes are running and since when?
- complete state: Clipboard, listening Sockets, TCP connections …

2005:

- Chris Betz - memparser
- George M. Garner Jr. and Robert-Jan Mora - kntlist
- Mariusz Burdach - *Windows Memory Forensics Toolkit* (WMFT) v0.1
Introduction.
Enumeration of Processes.

PsActive
ProcessHe

PsActive
ProcessHe

EPROCES
S
Introduction.
Direct Kernel Object Manipulation (DKOM).

PsActive
ProcessHe
ad

smrss

flink
blink

explorer

flink
blink

rk

flink
blink

Searching for Processes and Threads
Andreas Schuster
2006-08-14, page 5
Introduction.
Searching for Objects.

Why?

- Hidden objects are present in memory.
- Terminated objects may still be found in memory – for days!

2006:

- February: Aaron Wolters and Nick L. Petroni - FATkit
- March: PTfinder
- April: Harlan Carvey - Isproc
- May: Chris Carr - GREPEXEC (code not publicly available yet)
- June: Mariusz Burdach - WMFT v0.2
Searching for Processes and Threads.

Agenda.

1. Introduction

2. Searching for Objects
   2.1 Memory Allocations
   2.2 Kernel Object
   2.3 EPROCESS / ETHREAD

3. Proof of Concept – PTfinder

4. Conclusion

5. Questions & Answers
Searching for Objects.
Layers.

- memory allocation
- kernel object
- EPROCESS / ETHREAD
Searching for Objects.
Memory Allocation.

struct _POOL_HEADER, 9 elements, 0x8 bytes
+0x000 PreviousSize : UChar
+0x001 PoolIndex : UChar
+0x002 PoolType : UChar
+0x003 BlockSize : UChar
+0x004 PoolTag : Uint4B

PoolType: either one of the non-paged pool types or 0 („free block“)

BlockSize: constant for processes and threads, varies with OS version

PoolTag:
- Process: „Proc“
- Thread: „Thre“
- Protection flag (MSB) is set!
Searching for Objects.  
Kernel Objects.

struct __OBJECT_HEADER, 12 elements, 0x20 bytes
  +0x000  PointerCount    : Int4B
  +0x004  HandleCount     : Int4B
  +0x004  SEntry          : Ptr32
  +0x008  Type            : Ptr32 to struct __OBJECT_TYPE
  +0x00c  NameInfoOffset  : UChar
  +0x00d  HandleInfoOffset : UChar
  +0x00e  QuotaInfoOffset : UChar
  +0x00f  Flags           : UChar
  +0x010  ObjectCreateInfo : Ptr32
  +0x010  QuotaBlockCharged : Ptr32
  +0x014  SecurityDescriptor : Ptr32
  +0x018  Body
Searching for Objects.
Kernel Objects.

Type pointer depends on:
  ■ OS version
  ■ amount of main memory
  ■ other factors?

Values to scan for:
  ■ PsProcessType
  ■ PsThreadType
  ■ magic numer 0xbad0b0b0, indicates a defunct object
    (not necessarily a process or thread)

The object layer is not suitable to generate static signatures.
Searching for Objects.
EPROCESS / ETHREAD.

struct _EPROCESS, 94 elements, 0x290 bytes
+0x000 Pcb : struct _KPROCESS
  +0x000 Header : struct _DISPATCHER_HEADER
    +0x000 Type : 0x3
    +0x001 Absolute : 0
    +0x002 Size : 0x1b
    +0x003 Inserted : 0
    +0x004 SignalState : 0
    +0x008 WaitListHead : struct _LIST_ENTRY
...
  +0x070 LockEvent : struct _KEVENT
    +0x000 Header : struct _DISPATCHER_HEADER
...
  +0x130 WorkingSetLock : struct _FAST_MUTEX
    +0x000 Header : struct _DISPATCHER_HEADER

Similiar structures can also be found in ETHREAD.
Searching for Processes and Threads.

Agenda.

1. Introduction

2. Searching for Objects
   2.1 Memory Allocations
   2.2 Kernel Object
   2.3 EPROCESS / ETHREAD

3. Proof of Concept – PTfinder

4. Conclusion

5. Questions & Answers
PTfinder.
About the Tool.

PTfinder = Process and Thread finder

It’s just a proof of concept:
- small and simple
- used to experiment with signatures and output formats
- no conversion between physical and virtual addresses
- works on (almost) any dump file format

It is NOT meant to be a full memory forensics application.
PTfinder
Demo Environment.

- PTfinder
  [link](http://computer.forensikblog.de/files/ptfinder/ptfinder-current.zip)
- Perl
  [link](http://www.perl.org/)
- GraphViz
  [link](http://www.research.att.com/sw/tools/graphviz/)
- ZGRViewer
  [link](http://zvtm.sourceforge.net/zgrviewer.html)
- memory images from the DFRWS 2005 challenge
PTfinder.
List of Processes.

<table>
<thead>
<tr>
<th>No.</th>
<th>PID</th>
<th>TID</th>
<th>Time created</th>
<th>Time exited</th>
<th>Offset</th>
<th>PDB</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>622</td>
<td>324</td>
<td>2005-06-05</td>
<td>00:32:59</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>324</td>
<td>622</td>
<td>2005-06-05</td>
<td>14:09:27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>0</td>
<td>2005-06-05</td>
<td>00:55:08</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>668</td>
<td>8</td>
<td>2005-06-05</td>
<td>14:01:25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1412</td>
<td>7</td>
<td>2005-06-05</td>
<td>01:00:53</td>
<td>2005-06-05 01:00:53</td>
<td>0x001e1f60 VinMgmt.exe</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>216</td>
<td>1412</td>
<td>2005-06-05</td>
<td>00:32:44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>176</td>
<td>216</td>
<td>2005-06-05</td>
<td>03:36:31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>176</td>
<td>414</td>
<td>2005-06-05</td>
<td>01:25:54</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>164</td>
<td>176</td>
<td>2005-06-05</td>
<td>00:32:40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>164</td>
<td>2005-06-05</td>
<td>00:32:40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>168</td>
<td>100</td>
<td>2005-06-05</td>
<td>00:32:40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>156</td>
<td>168</td>
<td>2005-06-05</td>
<td>00:32:40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>152</td>
<td>156</td>
<td>2005-06-05</td>
<td>14:14:38</td>
<td>2005-06-05 14:14:38</td>
<td>0x012a3b2e amas.exe</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>592</td>
<td>152</td>
<td>2005-06-05</td>
<td>01:08:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1076</td>
<td>592</td>
<td>2005-06-05</td>
<td>00:35:10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>780</td>
<td>1076</td>
<td>2005-06-05</td>
<td>00:30:37</td>
<td>2005-06-05 00:30:37</td>
<td>0x012f3c9d metasploit</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>964</td>
<td>780</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>972</td>
<td>964</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>972</td>
<td>972</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>1000</td>
<td>972</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1012</td>
<td>1000</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1012</td>
<td>1012</td>
<td>2005-06-05</td>
<td>00:33:57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>888</td>
<td>1012</td>
<td>2005-06-05</td>
<td>00:33:52</td>
<td>2005-06-05 00:34:19</td>
<td>0x013f3509 userinit.exe</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>628</td>
<td>888</td>
<td>2005-06-05</td>
<td>00:33:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>628</td>
<td>628</td>
<td>2005-06-05</td>
<td>00:33:53</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>1048</td>
<td>628</td>
<td>2005-06-05</td>
<td>14:53:42</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>216</td>
<td>1048</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>216</td>
<td>216</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>760</td>
<td>216</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1064</td>
<td>760</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1072</td>
<td>1064</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>490</td>
<td>1072</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>490</td>
<td>490</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>436</td>
<td>490</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>436</td>
<td>436</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>464</td>
<td>436</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>464</td>
<td>464</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>1132</td>
<td>464</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>1132</td>
<td>1132</td>
<td>2005-06-05</td>
<td>00:33:48</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Searching for Processes and Threads
Andreas Schuster
2006-08-14, page 16
PTfinder.
Building the Process Tree.
PTfinder.
Close-up.
PTfinder.
Analyzing the Incident – LSASS Exploit.

LSASS.EXE (Local Security Authority Subsystem) is not expected to spawn processes.

Metasploit.exe indicates the usage of a well-known exploit construction kit of the same name.

Further examination shows that UMGR32.EXE is an instance of BackOrifice by Cult of the Dead Cow.
PTfinder.
Analyzing the Incident – Trojan Horse.

A process „dfrws2005.exe“ is launched by the trojan horse.
The process terminates within a second. It does not report an error.
PTfinder.
Searching the Graph.
PTfinder.
Rootkit and Backdoor Service.
PTfinder.
Persistance through a Reboot.

Processes appearing to be started prior to system boot (1st image):

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Image Name</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-06-03</td>
<td>01:25:53Z</td>
<td>csrss.exe</td>
<td>168</td>
</tr>
<tr>
<td>2005-06-03</td>
<td>01:25:54Z</td>
<td>winlogon.exe</td>
<td>164</td>
</tr>
</tbody>
</table>

2nd image:

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Image Name</th>
<th>PID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2005-06-03</td>
<td>01:25:53Z</td>
<td>csrss.exe</td>
<td>168</td>
</tr>
<tr>
<td>2005-06-05</td>
<td>00:32:40Z</td>
<td>smss.exe</td>
<td>156</td>
</tr>
<tr>
<td>2005-06-05</td>
<td>00:32:43Z</td>
<td>csrss.exe</td>
<td>180</td>
</tr>
</tbody>
</table>
PTfinder.
Reliability.

Setup:

- memory dumps obtained from clean installations of Microsoft Windows XP, XP SP1, XP SP2 and Windows Server 2003
- lists of processes and threads produced by PTfinder and Microsoft kernel debugger (kd, windbg) and then compared

Results:

- False negatives: PTfinder did not miss any process/thread shown by kd. No false negatives.
- False positives: PTfinder shows some processes and threads not listed by kd. They all appear to be valid, with some artifacts from a prior run of Windows. So no false positives.
Searching for Processes and Threads.
Agenda.

1. Introduction

2. Searching for Objects
   2.1 Memory Allocations
   2.2 Kernel Object
   2.3 EPROCESS / ETHREAD

3. Proof of Concept – PTfinder

4. Conclusion

5. Questions & Answers
Searching for Processes and Threads.
Conclusion.

Results:
- works on raw dumps (dd), Windows crash dumps (DMP) and VMware (4.x/5.x) suspended sessions (VMSS/VMEM)
- reliably finds active processes and threads as well as traces of defunct ones

Future work:
- adopt signatures to Microsoft Vista/Longhorn
- evaluate possibilities to by-pass signatures
Searching for Processes and Threads.

Agenda.

1. Introduction

2. Searching for Objects
   2.1 Memory Allocations
   2.2 Kernel Object
   2.3 EPROCESS / ETHREAD

3. Proof of Concept – PTfinder

4. Conclusion

5. Questions & Answers
Thank You for Your Attention.

Andreas Schuster
Deutsche Telekom AG
Group Security
andreas.schuster@telekom.de