



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
e

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?

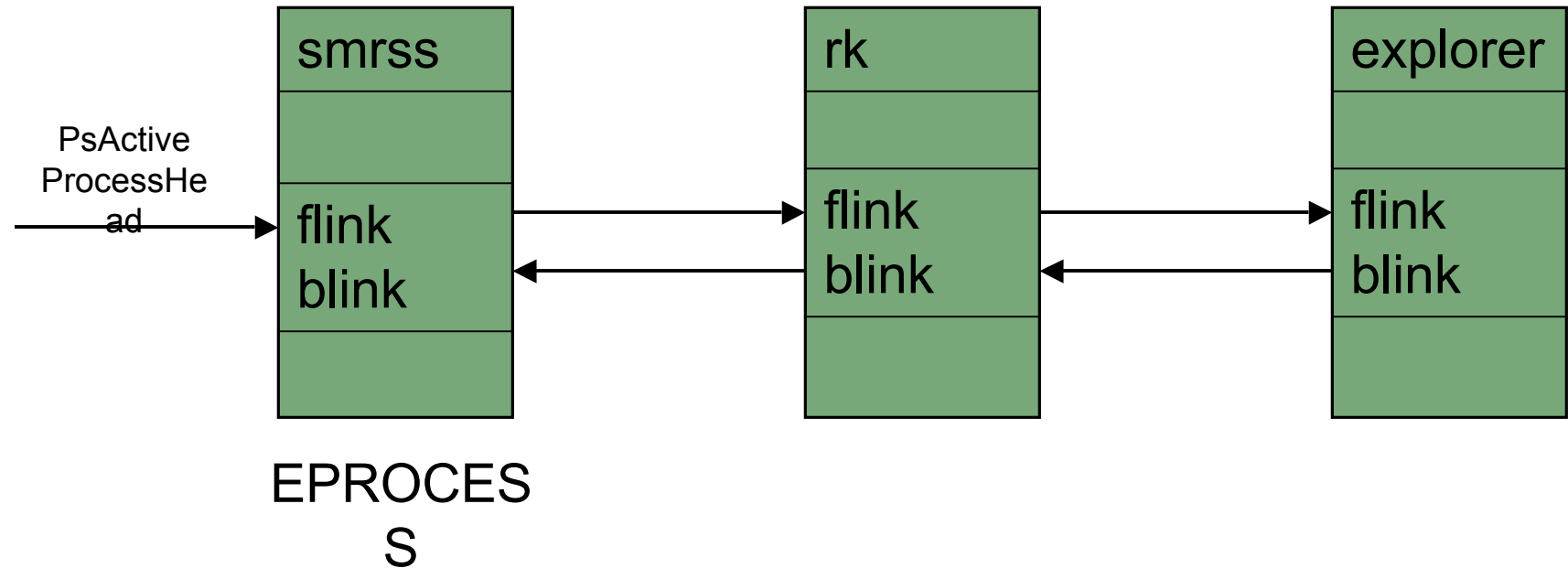
- certain attacks don't leave traces on disk
- Which processes are running and since when?
- complete state: Clipboard, listening Sockets, TCP connections ...

2005:

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

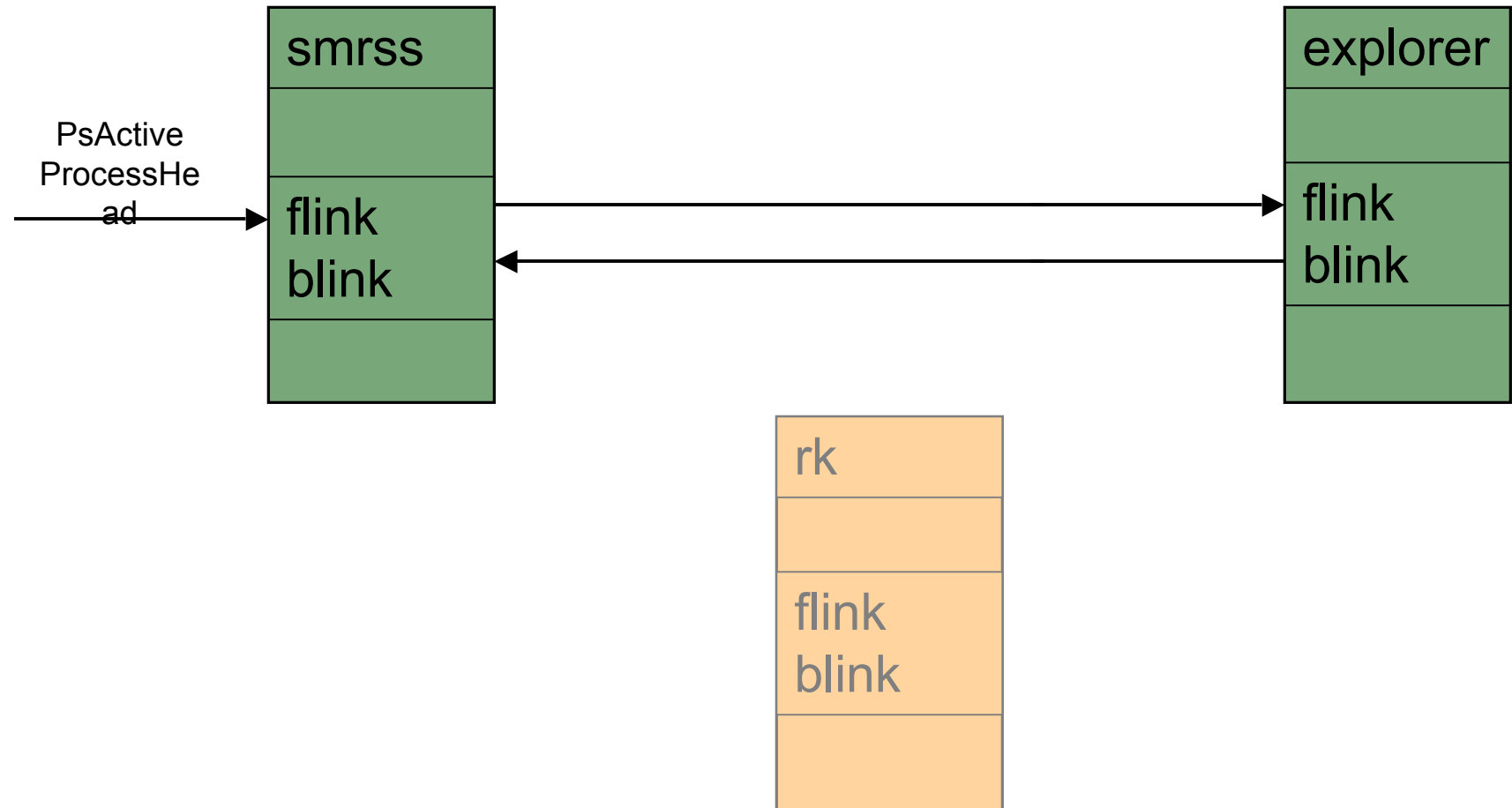
Introduction.

Enumeration of Processes.



Introduction.

Direct Kernel Object Manipulation (DKOM).



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.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E1:FB30h:	2C	CB	1C	FF	00	00	00	00	00	00	00	00	00	00	00	00																
E1:FB40h:	04	80	01	16	50	72	6F	E3	02	00	00	00	01	00	00	00																
E1:FB50h:	60	51	E2	FC	00	00	00	20	20	B6	46	80	78	0C	00	E1																
E1:FB60h:	03	00	1B	00	01	00	00	00	68	CB	1C	FF	68	CB	1C	FF																
E1:FB70h:	70	CB	1C	FF	70	CB	1C	FF	00	80	C9	06	00	90	05	07																
E1:FB80h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																
E1:FD30h:	04	80	00	00	00	00	00	00	00	00	00	00	00	00	00	00																
E1:FD40h:	E8	07	E0	FC	00	00	00	00	48	CD	1C	FF	48	CD	1C	FF																
E1:FD50h:	00	00	00	00	00	00	00	00	00	00	00	00	64	66	72	77																
E1:FD60h:	73	32	30	30	35	2E	65	78	65	00	00	00	00	00	00	00																
E1:FD70h:	00	02	00	04	00	00	00	00	00	00	00	00	00	00	00	00																
E1:FD80h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00																
E1:FD90h:	00	00	00	00	00	00	00	00	0A	00	00	00	00	00	00	00																
E1:FDA0h:	03	00	00	00	00	00	00	00	26	00	00	00	00	00	00	00																
E1:FDB0h:	22	04	00	00	00	00	00	00	D8	00	00	00	00	00	00	00																

 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
```

Similar 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

<http://computer.forensikblog.de/files/ptfinder/ptfinder-current.zip>

- Perl

<http://www.perl.org/>

- GraphViz

<http://www.research.att.com/sw/tools/graphviz/>

- ZGRViewer

<http://zvtm.sourceforge.net/zgrviewer.html>

- memory images from the DFRWS 2005 challenge

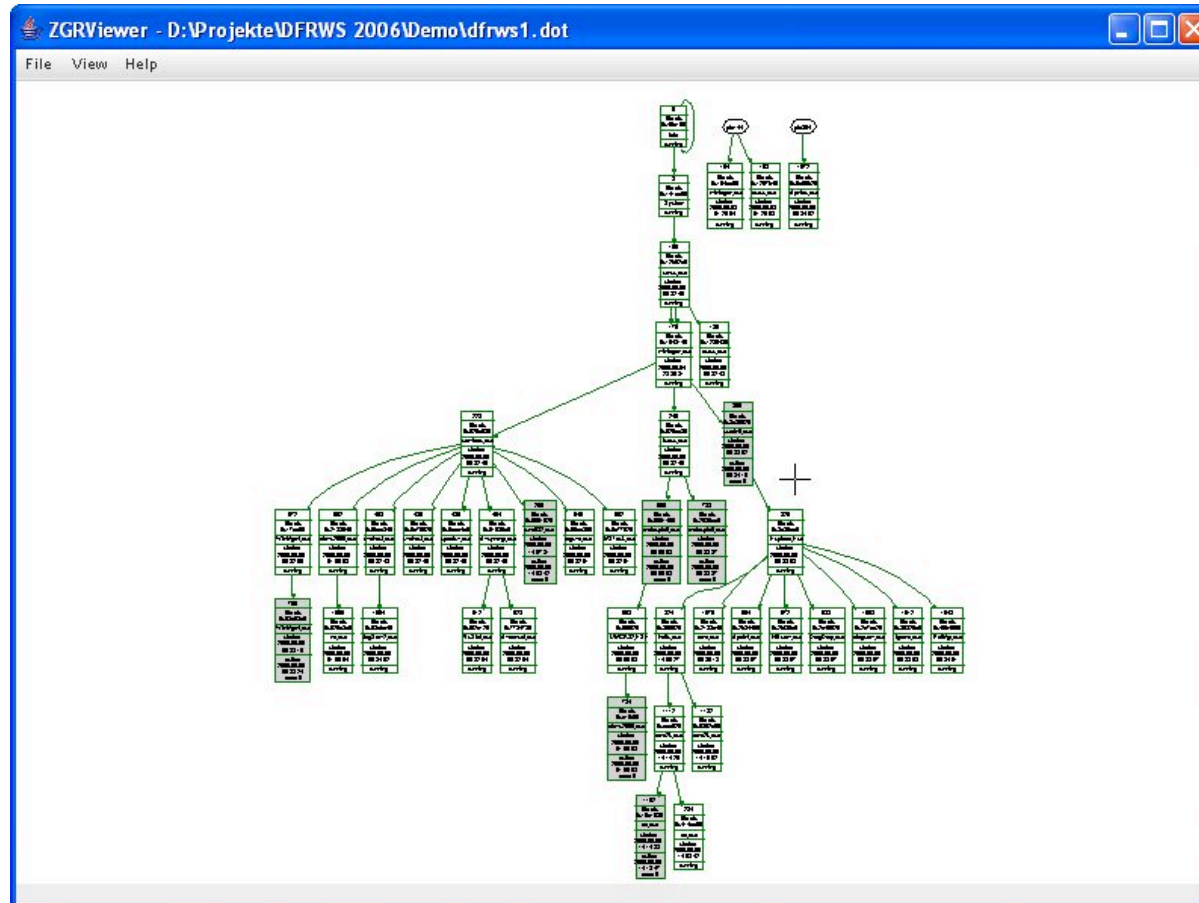
PTfinder.

List of Processes.

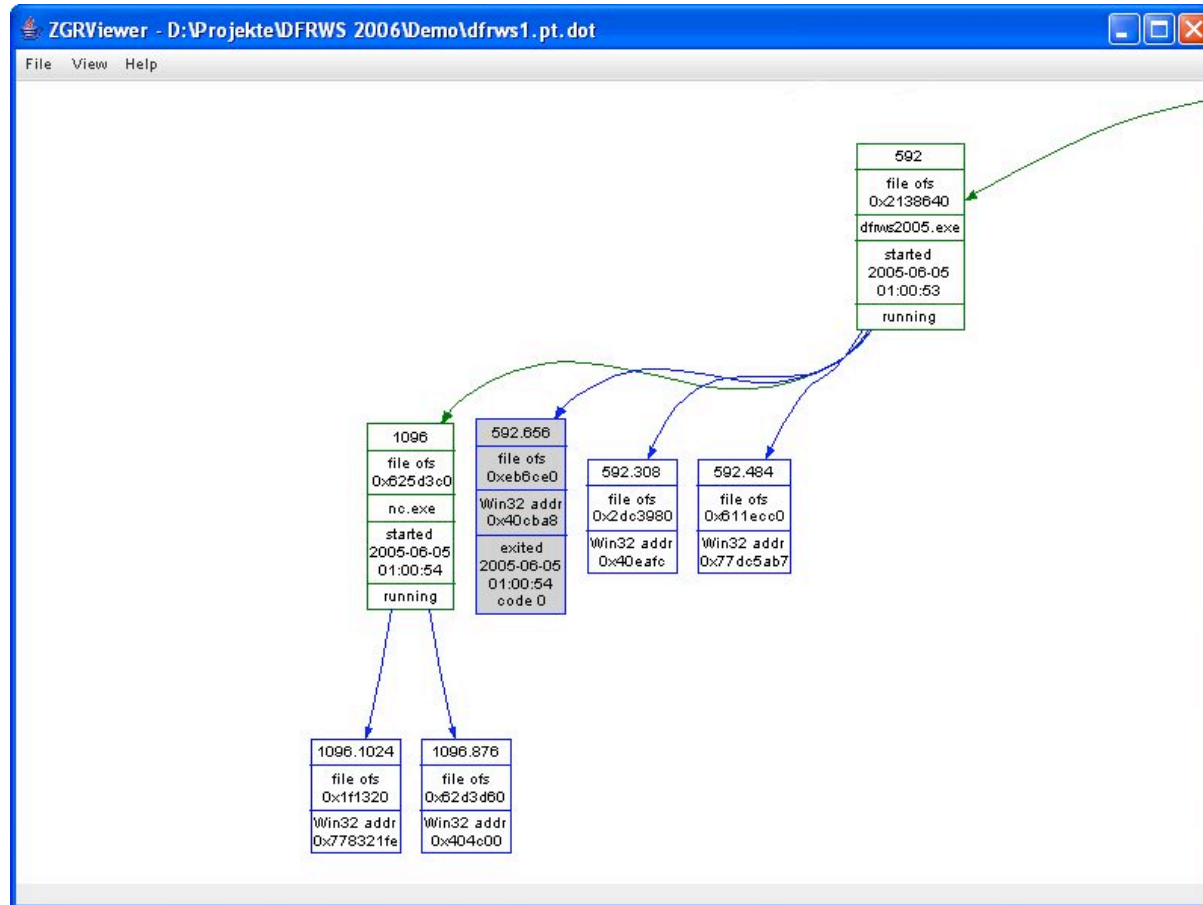
Shell - ptfinder_w2k.pl --nothreads dfrws2005-physical-memory1.ddmp

No.	Type	PID	TID	Time created	Time exited	Offset	PDB	Remarks
1	Proc	672		2005-06-05 00:32:59		0x0017dd60	0x01a24000	WinMgmt.exe
2	Proc	324		2005-06-05 14:09:27		0x00306020	0x06f53000	helix.exe
3	Proc	0				0x0046d160	0x00030000	Idle
4	Proc	668		2005-06-05 00:55:08		0x0095f020	0x075a7000	UMGR32.EXE
5	Proc	1112		2005-06-05 14:14:25		0x00dcc020	0x039a2000	cmd2k.exe
6	Proc	784		2005-06-05 01:00:53	2005-06-05 01:00:53	0x00e1fb60	0x06c98000	dfrws2005.e
7	Proc	176		2005-06-05 00:32:44		0x01045d60	0x04fe4000	winlogon.ex
8	Proc	176		2005-06-04 23:36:31		0x01048140	0x04fc4000	winlogon.ex
9	Proc	164		2005-06-03 01:25:54		0x0104ca00	0x04f24000	winlogon.ex
10	Proc	180		2005-06-05 00:32:43		0x01286480	0x0429f000	csrss.exe
11	Proc	168		2005-06-03 01:25:53		0x01297b40	0x041df000	csrss.exe
12	Proc	156		2005-06-05 00:32:40		0x012b62c0	0x03104000	smss.exe
13	Proc	8				0x0141dc60	0x00030000	System
14	Proc	1152		2005-06-05 14:14:38	2005-06-05 14:18:47	0x019d1980	0x03bcf000	dd.exe
15	Proc	592		2005-06-05 01:00:53		0x02138640	0x03e02000	dfrws2005.e
16	Proc	1076		2005-06-05 00:35:18		0x02138c40	0x0575e000	cmd.exe
17	Proc	788		2005-06-05 00:38:37	2005-06-05 00:38:37	0x02686cc0	0x07234000	metasploit.
18	Proc	964		2005-06-05 00:33:57		0x02b84400	0x02bf1000	Apntex.exe
19	Proc	972		2005-06-05 00:33:57		0x02bf86e0	0x02ce7000	Hkserve.exe
20	Proc	988		2005-06-05 00:33:57		0x02c46020	0x02dbc000	DragDrop.ex
21	Proc	1008		2005-06-05 00:33:57		0x02e7ea20	0x02e4c000	alogserv.ex
22	Proc	1012		2005-06-05 00:33:58		0x030826a0	0x02ce5000	tgcmd.exe
23	Proc	800		2005-06-05 00:33:52	2005-06-05 00:34:19	0x03e35020	0x03fb0000	userinit.ex
24	Proc	820		2005-06-05 00:33:53		0x03e35ae0	0x03ca1000	Explorer.Ex
25	Proc	1048		2005-06-05 00:34:01		0x040b4660	0x043de000	PcfMgr.exe
26	Proc	284		2005-06-05 14:53:42		0x0414dd60	0x01d9e000	dd.exe
27	Proc	228		2005-06-05 00:32:45		0x0520a080	0x052e5000	services.ex
28	Proc	240		2005-06-05 00:32:45		0x0529ea80	0x052ad000	lsass.exe
29	Proc	760		2005-06-05 00:33:10	2005-06-05 00:33:24	0x058a08a0	0x0059e000	WinMgmt.exe
30	Proc	1064		2005-06-05 00:34:02		0x058cbc40	0x05a23000	JogServ2.ex
31	Proc	1072		2005-06-05 00:34:02		0x05a59b20	0x05d0c000	Apntex.exe
32	Proc	408		2005-06-05 00:32:48		0x05bec840	0x05cb4000	svchost.exe
33	Proc	480		2005-06-05 00:32:49		0x05c70020	0x05f78000	svchost.exe
34	Proc	436		2005-06-05 00:32:49		0x05dce4c0	0x05e2e000	spoolsv.exe
35	Proc	464		2005-06-05 00:32:49		0x061539e0	0x06173000	Avsynmgr.ex
36	Proc	1096		2005-06-05 01:00:54		0x0625d3c0	0x0600d000	nc.exe
37	Proc	1132		2005-06-05 14:10:52		0x06352d60	0x058dd000	cmd2k.exe
38	Proc	296		2005-06-05 14:07:31	2005-06-05 14:08:42	0x06601020	0x07cd5000	rundll32.ex
39	Proc	600		2005-06-05 00:55:08	2005-06-05 00:55:08	0x06601460	0x01a6d000	metasploit.

PTfinder. Building the Process Tree.

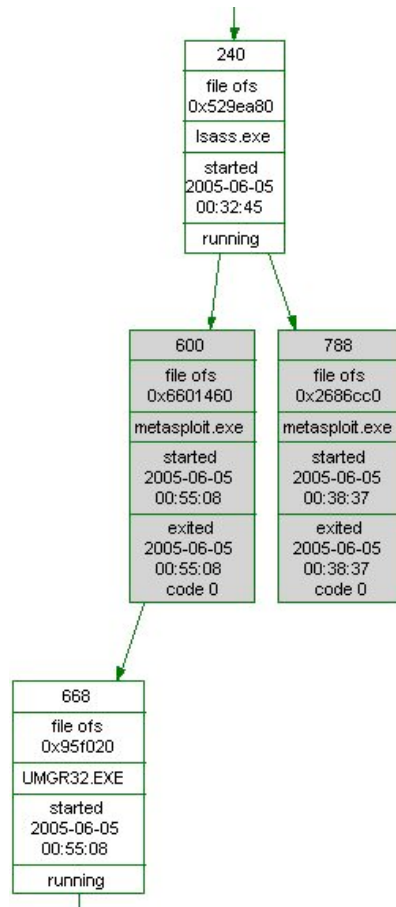


PTfinder. Close-up.



PTfinder.

Analyzing the Incident – LSASS Exploit.



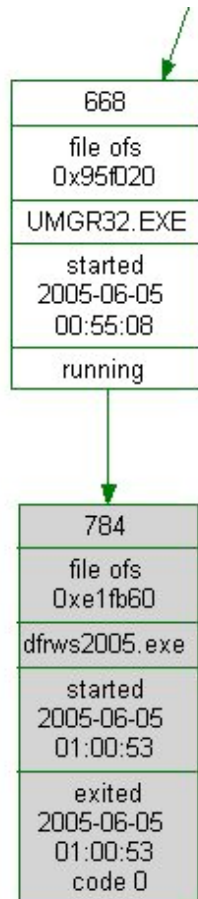
LSASS.EXE (Local Security Authority Subsystem) is not expected to spawn processes.

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.

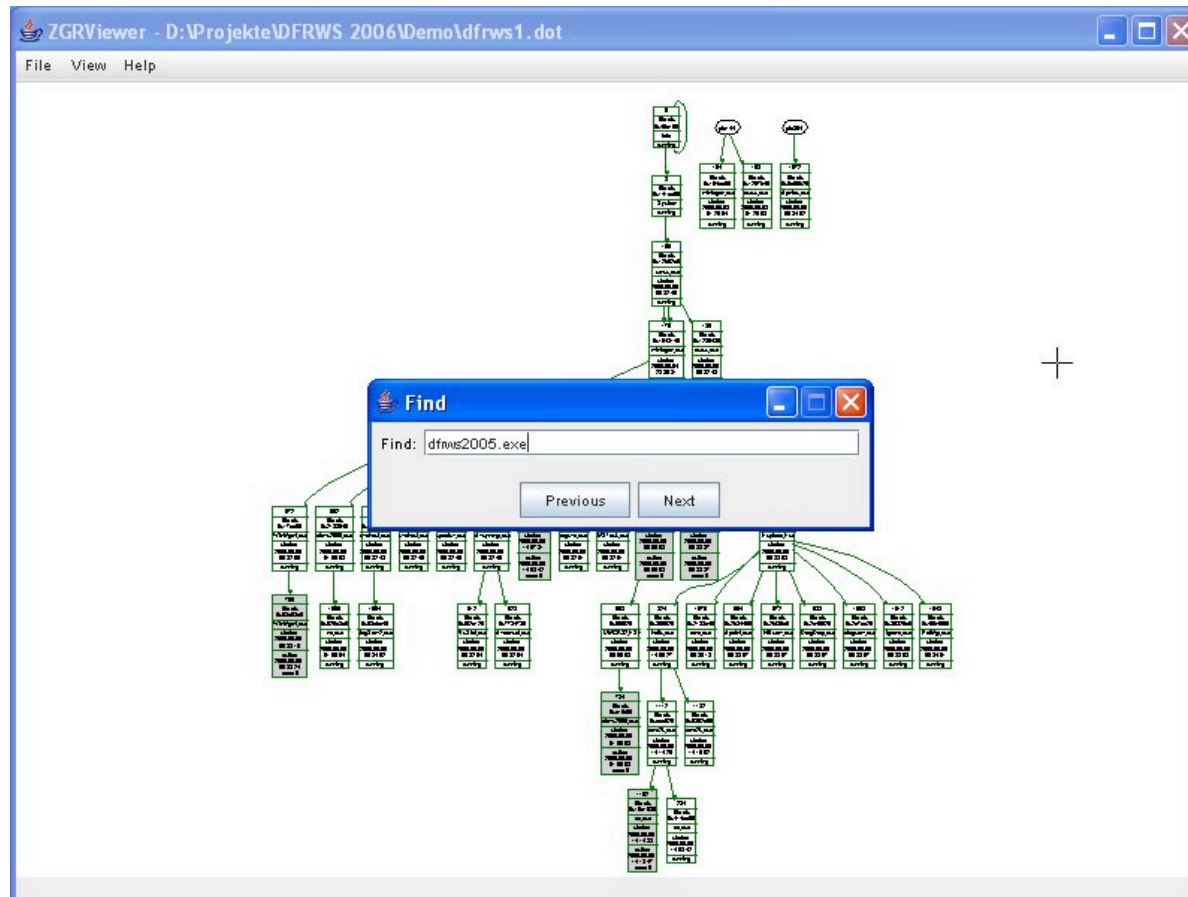
Analyzing the Incident – Trojan Horse.



A process „dfrws2005.exe“ is launched by the trojan horse.

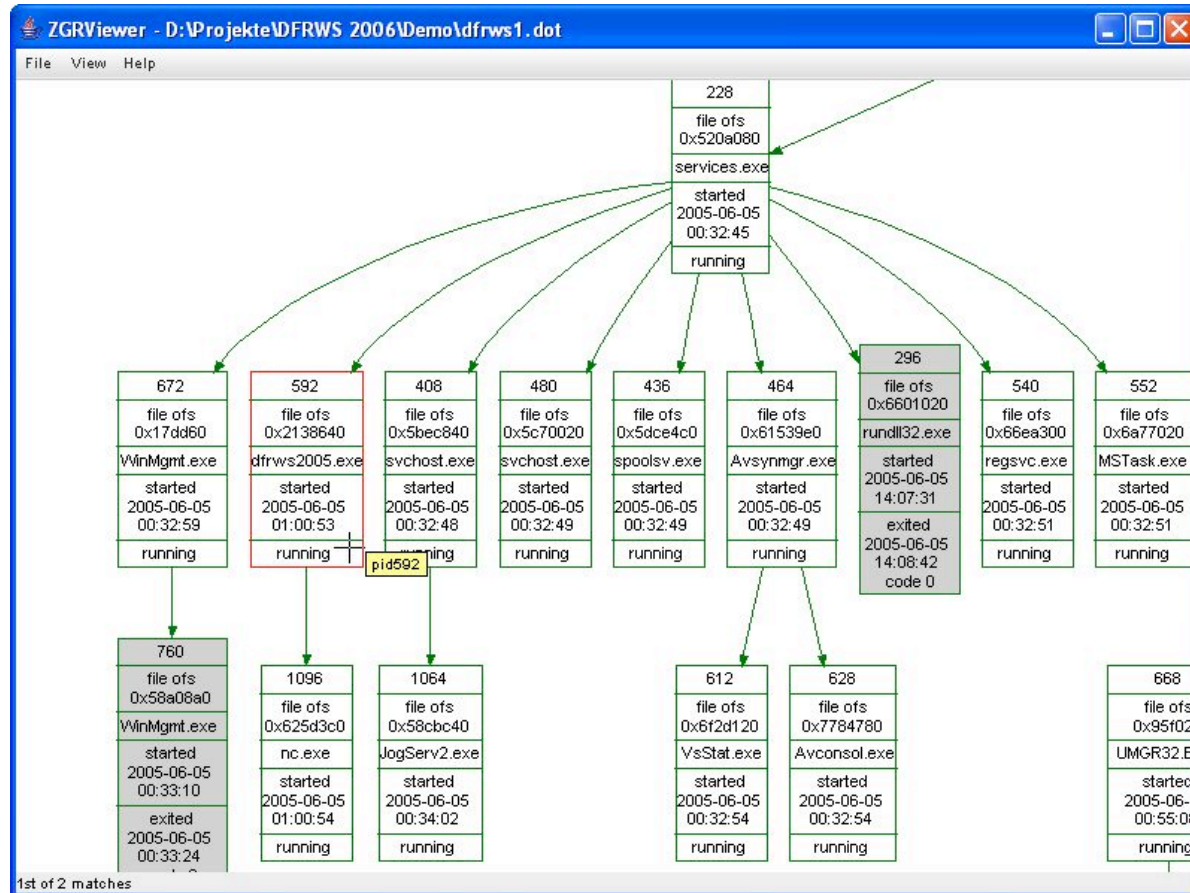
The process terminates within a second. It does not report an error.

PTfinder. Searching the Graph.



PTfinder.

Rootkit and Backdoor Service.



PTfinder.

Persistence through a Reboot.

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

<u>Date</u>	<u>Time</u>	<u>Image Name</u>	<u>PID</u>
2005-06-03	01:25:53Z	csrss.exe	168
2005-06-03	01:25:54Z	winlogon.exe	164
2005-06-04	23:36:31Z	winlogon.exe	176

2nd image:

<u>Date</u>	<u>Time</u>	<u>Image Name</u>	<u>PID</u>
2005-06-03	01:25:53Z	csrss.exe	168
2005-06-05	00:32:40Z	smss.exe	156
2005-06-05	00:32:43Z	csrss.exe	180

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.d](mailto:andreas.schuster@telekom.de)
e