# PV204 Security technologies

## In-Memory Malware Analysis

Václav Lorenc
Senior Security Analyst, Oracle + NetSuite

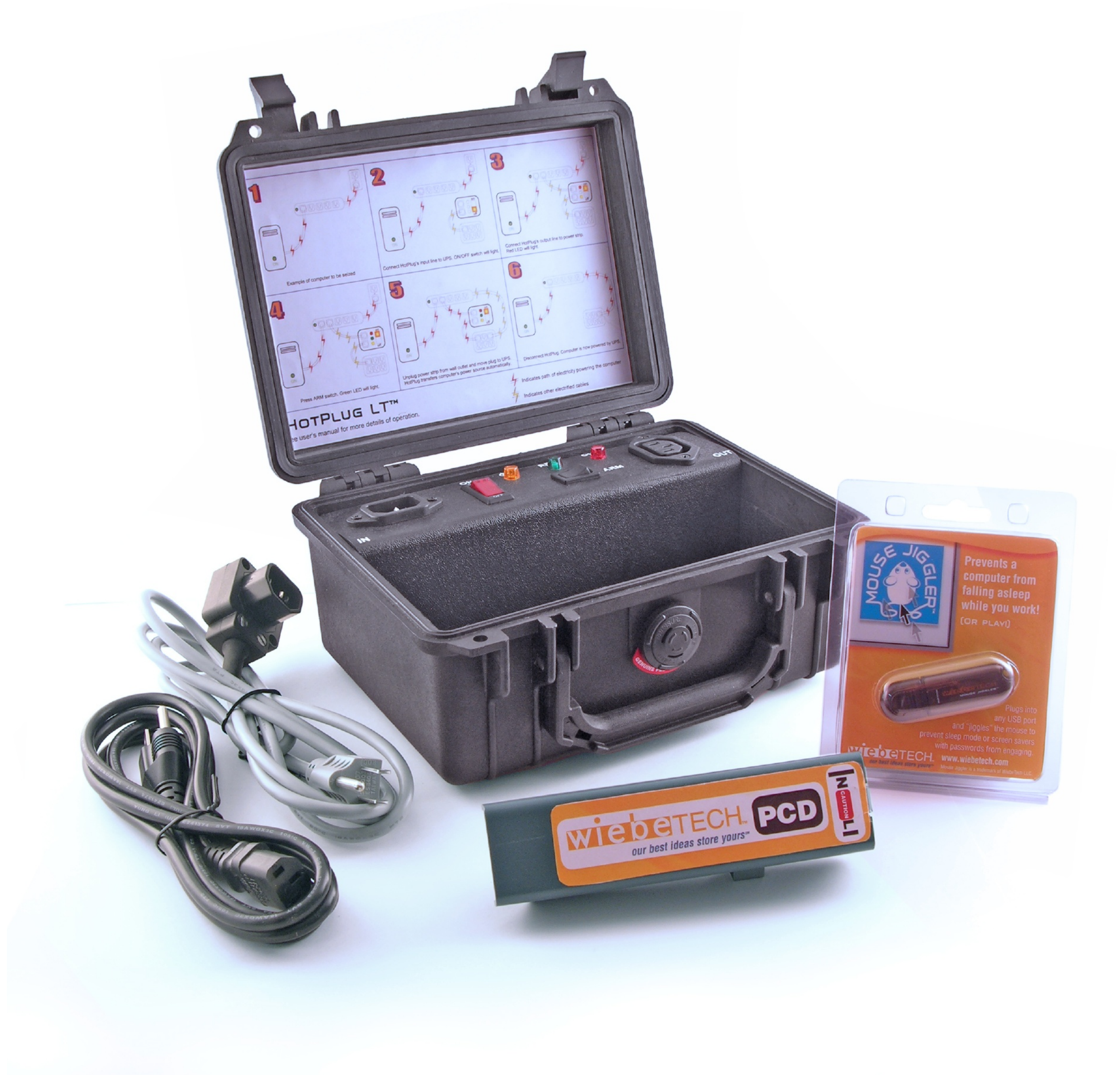
CROCS
Centre for Research on
Cryptography and Security

# Agenda

- Basic intro
  - No assembly required
  - No malware (de)obfuscation magic
- How does the OS look “inside”?
  - Processes and other data structures
  - How the memory is organized
- Common tools used for analysis
- Searching for system “oddities”
  - What are the important system indicators?
- Real samples discussed and analyzed! (Labs)

# Why memory analysis?

- **It's fun!**
- Acquiring evidence for legal investigations
  - It used to be different in the past
- Incident response activities
  - Easy way how to learn more about the attackers
  - Malicious binary may only be present in memory
- Technical simplification of reverse engineering
  - No binary obfuscation present – the code has to run


HOTPLUG LT™
...e user's manual for more details of operation.
1
2
3
4
5
6
IN
OUT
ARM
MOUSE JIGGLER
Prevents a computer from falling asleep while you work!
(OR PLAY!)
Plugs into any USB port and "jiggles" the mouse to prevent sleep mode or screen savers with passwords from engaging.
wiebetech
www.wiebetech.com
wiebeTECH PCD
our best ideas store yours™
CAUTION

# Challenges in Reverse Engineering (RE)

- Assembly language (for multiple platforms)
  - Plus undocumented instructions (or behavior)
- Anti-debugging tricks
  - Exceptions, interrupts, PE manipulations, time checking, ...
- Anti-VM tricks
  - Uncommon behavior of known instructions
  - Registry detections, HW detections
- Code obfuscation/packing
  - The most challenging to overcome, mostly

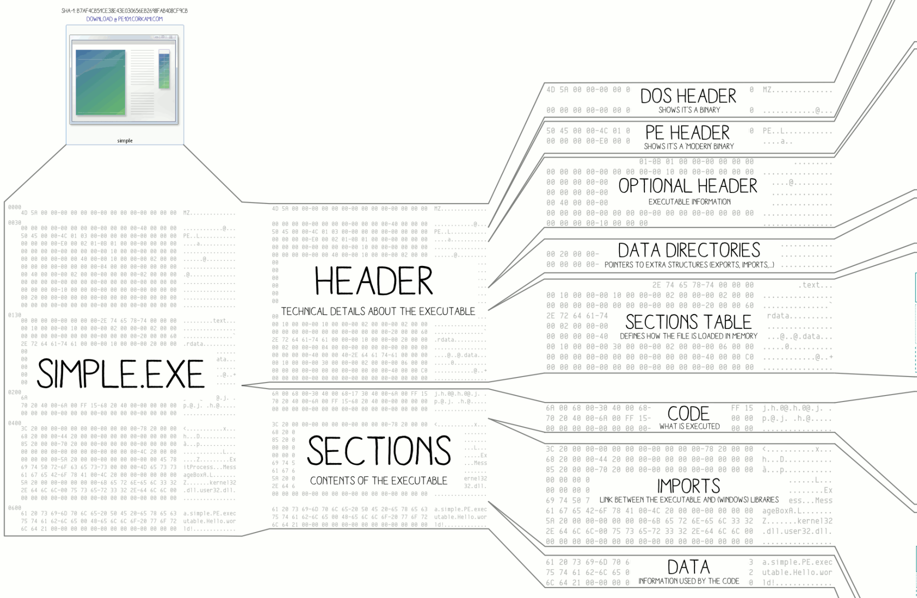
# PE File Format

PE[101] PORTABLE EXECUTABLE a windows executable walkthrough

V2.0L, 5TH DECEMBER 2012
CREATIVE COMMONS 3.0 BY

ANGE ALBERTINI
CORKAMI.COM

## DISSECTED PE


SIMPLE.EXE
HEADER
TECHNICAL DETAILS ABOUT THE EXECUTABLE
SECTIONS
CONTENTS OF THE EXECUTABLE
DOS HEADER
SHOWS IT'S A BINARY
PE HEADER
SHOWS IT'S A 'MODERN' BINARY
OPTIONAL HEADER
EXECUTABLE INFORMATION
DATA DIRECTORIES
POINTERS TO EXTRA STRUCTURES (EXPORTS, IMPORTS,...)
SECTIONS TABLE
DEFINES HOW THE FILE IS LOADED IN MEMORY
CODE
WHAT IS EXECUTED
IMPORTS
LINK BETWEEN THE EXECUTABLE AND (WINDOWS) LIBRARIES
DATA
INFORMATION USED BY THE CODE
simple

HEXADECIMAL DUMP / ASCII DUMP / FIELDS / VALUES / EXPLANATION

```
4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ..............
Offset:0x30
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 00  ............@...
```

| FIELDS | VALUES | EXPLANATION |
|---|---|---|
| e_magic | 'MZ' | CONSTANT SIGNATURE |
| e_lfanew | 0x40 | OFFSET OF THE PE HEADER 1 |

```
Offset:0x40
50 45 00 00-4C 01 03 00-00 00 00 00-00 00 00 00  PE..L...........
00 00 00 00-E0 00 02 01...                       ....a...
```

| FIELDS | VALUES | EXPLANATION |
|---|---|---|
| Signature | 'PE', 0, 0 | CONSTANT SIGNATURE |
| Machine | 0x14c [intel 386] | PROCESSOR: ARM/MIPS/INTEL/... |
| NumberOfSections | 3 | NUMBER OF SECTIONS 2 |
| SizeOfOptionalHeader | 0xe0 | RELATIVE OFFSET OF THE SECTION TABLE 2 |
| Characteristics | 0x102 [32b EXE] | EXE/DLL/... |

```
                  Offset:0x58
             ...0B 01 00 00-00 00 00 00                 ........
00 00 00 00-00 00 00 00-00 10 00 00-00 00 00 00  ................
00 00 00 00-00 00 40 00-00 10 00 00-00 02 00 00  ......@.........
00 00 00 00-00 00 00 00-04 00 00 00-00 00 00 00  ................
00 40 00 00-00 02 00 00-00 00 00 00-02 00 00 00  .@..............
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ................
00 00 00 00-10 00 00 00...                       ........
```

| FIELDS | VALUES | EXPLANATION |
|---|---|---|
| Magic | 0x10b [32b] | 32 BITS/64 BITS |
| AddressOfEntryPoint | 0x1000 | WHERE EXECUTION STARTS 5 |
| ImageBase | 0x400000 | ADDRESS WHERE THE FILE SHOULD BE MAPPED IN MEMORY 3 |
| SectionAlignment | 0x1000 | WHERE SECTIONS SHOULD START IN MEMORY 2 |
| FileAlignment | 0x200 | WHERE SECTIONS SHOULD START ON FILE 2 |
| MajorSubsystemVersion | 4 [NT 4 or later] | REQUIRED VERSION OF WINDOWS |
| SizeOfImage | 0x4000 | TOTAL MEMORY SPACE REQUIRED |
| SizeOfHeaders | 0x200 | TOTAL SIZE OF THE HEADERS 3 |
| Subsystem | 2 [GUI] | DRIVER/GRAPHICAL/COMMAND LINE/... |
| NumberOfRvaAndSizes | 16 | NUMBER OF DATA DIRECTORIES 4 |

```
             ...00 00 00 00-00 00 00 00          ........
00 20 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ................
00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00  ................
```

| FIELDS | VALUES | EXPLANATION |
|---|---|---|
| ImportsVA | 0x2000 | RVA* OF THE IMPORTS 4 |

```
                  Offset:0x138
                2E 74 65 78-74 00 00 00          .text...
00 10 00 00-00 10 00 00-00 02 00 00-00 02 00 00  ................
00 00 00 00-00 00 00 00-00 00 00 00-20 00 00 60  ...............`
2E 72 64 61-74 61 00 00-00 10 00 00-00 20 00 00  .rdata..........
00 02 00 00-00 04 00 00-00 00 00 00-00 00 00 00  ................
00 00 00 00-40 00 00 40-2E 64 61 74-61 00 00 00  ....@..@.data...
00 10 00 00-00 30 00 00-00 02 00 00-00 06 00 00  .....0..........
00 00 00 00-00 00 00 00-00 00 00 00-40 00 00 C0  ............@..+
```

### SECTIONS TABLE

| NAME | VIRTUALSIZE (RVA*) | VIRTUALADDRESS (RVA*) | SIZEOFRAWDATA (PHYSICAL SIZE) | POINTERTORAWDATA (PHYSICAL OFFSET) | CHARACTERISTICS |
|---|---|---|---|---|---|
| .text | 0x1000 | 0x1000 | 0x200 | 0x200 | CODE EXECUTE READ |
| .rdata | 0x1000 | 0x2000 | 0x200 | 0x400 | INITIALIZED READ |
| .data | 0x1000 | 0x3000 | 0x200 | 0x600 | DATA READ WRITE |

FOR EACH SECTION, A SIZEOFRAWDATA SIZED BLOCK IS READ FROM THE FILE AT POINTERTORAWDATA OFFSET.
IT WILL BE LOADED IN MEMORY AT ADDRESS IMAGEBASE + VIRTUALADDRESS IN A VIRTUALSIZE SIZED BLOCK, WITH SPECIFIC CHARACTERISTICS.

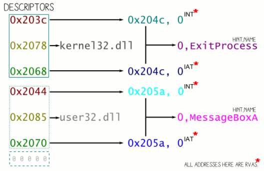
```
Offset:0x200/RVA:0x401000
6A 00 68 00-30 40 00 68-17 30 40 00-6A 00 FF 15  j.h.0@.h.0@.j. .
70 20 40 00-6A 00 FF 15-68 20 40 00              p.@.j. .h.@.
```

X86 ASSEMBLY / EQUIVALENT C CODE

```
push 0
push 0x403000
push 0x403017
push 0
call [0x402070]        MessageBox(0, "Hello World!","a simple PE executable", 0);
push 0
call [0x402068]        ExitProcess(0);
```

```
Offset:0x400/RVA:0x402000
3C 20 00 00-00 00 00 00-00 00 00 00-78 20 00 00  <...........x...
68 20 00 00-44 20 00 00-00 00 00 00-00 00 00 00  h...D...........
85 20 00 00-70 20 00 00-00 00 00 00-00 00 00 00  à...p...........
00 00 00 00-00 00 00 00-00 00 00 00-4C 20 00 00  ............L...
00 00 00 00-5A 20 00 00-00 00 00 00-00 00 45 78  ....Z.........Ex
69 74 50 72-6F 63 65 73-73 00 00 00-4D 65 73 73  itProcess...Mess
61 67 65 42-6F 78 41 00-4C 20 00 00-00 00 00 00  ageBoxA.L.......
5A 20 00 00-00 00 00 00-6B 65 72 6E-65 6C 33 32  Z.......kernel32
2E 64 6C 6C-00 75 73 65-72 33 32 2E-64 6C 6C 00  .dll.user32.dll.
```

### IMPORTS STRUCTURES



### CONSEQUENCES

AFTER LOADING,
0X402068 WILL POINT TO KERNEL32.DLL`S EXITPROCESS
0X402070 WILL POINT TO USER32.DLL`S MESSAGEBOXA

```
Offset:0x600/RVA:0x403000
61 20 73 69-6D 70 6C 65-20 50 45 20-65 78 65 63  a.simple.PE.exec
75 74 61 62-6C 65 00 48-65 6C 6C 6F-20 77 6F 72  utable.Hello.wor
6C 64 21 00                                      ld!.
```

STRINGS

```
a simple PE executable\0
Hello world!\0
```

THIS IS THE WHOLE FILE, HOWEVER, MOST PE FILES CONTAIN MORE ELEMENTS.
EXPLANATIONS ARE SIMPLIFIED, FOR CONCISENESS.

## LOADING PROCESS

### 1 HEADERS

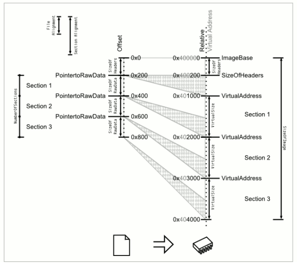
THE *DOS HEADER* IS PARSED
THE *PE HEADER* IS PARSED
(ITS OFFSET IS *DOS HEADER*`S E_LFANEW)
THE *OPTIONAL HEADER* IS PARSED
(IT FOLLOWS THE *PE HEADER*)

### 2 SECTIONS TABLE

SECTIONS TABLE IS PARSED
(IT IS LOCATED AT: OFFSET (*OPTIONALHEADER*) + *SIZEOFOPTIONALHEADER*)
IT CONTAINS *NUMBEROFSECTIONS* ELEMENTS
IT IS CHECKED FOR VALIDITY WITH ALIGNMENTS:
*FILEALIGNMENTS* AND *SECTIONALIGNMENTS*

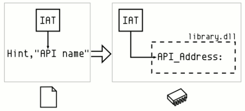
### 3 MAPPING

THE FILE IS MAPPED IN MEMORY ACCORDING TO:
THE *IMAGEBASE*
THE *SIZEOFHEADERS*
THE SECTIONS TABLE



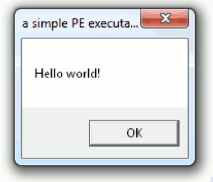
### 4 IMPORTS

*DATADIRECTORIES* ARE PARSED
THEY FOLLOW THE *OPTIONALHEADER*
THEIR NUMBER IS *NUMOFRVAANDSIZES*
*IMPORTS* ARE ALWAYS #2
*IMPORTS* ARE PARSED
EACH DESCRIPTOR SPECIFIES A *DLLNAME*
THIS DLL IS LOADED IN MEMORY
*IAT* AND *INT* ARE PARSED SIMULTANEOUSLY
FOR EACH API IN *INT*
ITS ADDRESS IS WRITTEN IN THE *IAT* ENTRY



### 5 EXECUTION

CODE IS CALLED AT THE *ENTRYPOINT*
THE CALLS OF THE CODE GO VIA THE IAT TO THE APIS


a simple PE executa...

Hello world!

## NOTES

MZ HEADER *AKA* DOS_HEADER
STARTS WITH 'MZ' (INITIALS OF *MARK ZBIKOWSKI* MS-DOS DEVELOPER)

PE HEADER *AKA* IMAGE_FILE_HEADERS / COFF FILE HEADER
STARTS WITH 'PE' (PORTABLE EXECUTABLE)

OPTIONAL HEADER *AKA* IMAGE_OPTIONAL_HEADER
OPTIONAL ONLY FOR NON-STANDARD PES BUT REQUIRED FOR EXECUTABLES

RVA RELATIVE VIRTUAL ADDRESS
ADDRESS RELATIVE TO IMAGEBASE (AT IMAGEBASE, RVA = 0)
ALMOST ALL ADDRESSES OF THE HEADERS ARE RVAS
IN CODE, ADDRESSES ARE *NOT* RELATIVE.

INT IMPORT NAME TABLE
NULL-TERMINATED LIST OF POINTERS TO HINT, NAME STRUCTURES

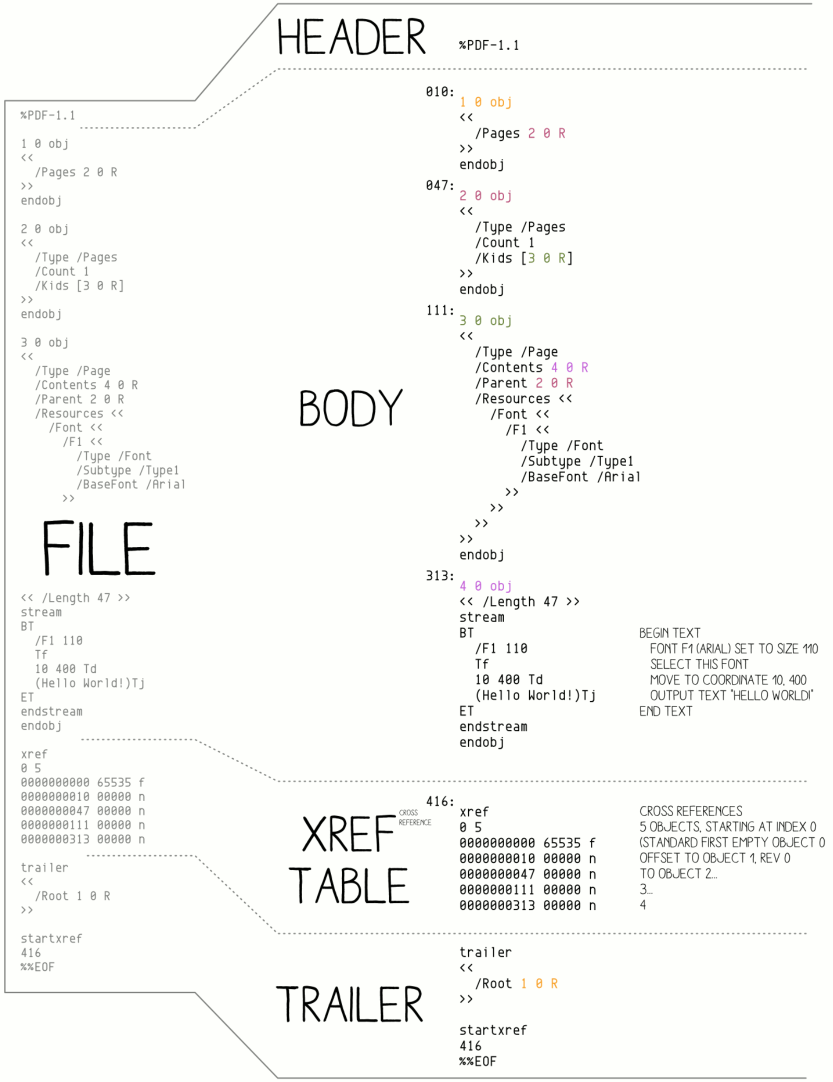
IAT IMPORT ADDRESS TABLE
NULL-TERMINATED LIST OF POINTERS
ON FILE IT IS A COPY OF THE INT
AFTER LOADING IT POINTS TO THE IMPORTED APIS

HINT
INDEX IN THE EXPORTS TABLE OF A DLL TO BE IMPORTED
NOT REQUIRED BUT PROVIDES A SPEED-UP BY REDUCING LOOK-UP

# PDF101 an Adobe document walkthrough

PORTABLE DOCUMENT FORMAT

ANGE ALBERTINI
CORKAMI.COM



FILE

```
%PDF-1.1

1 0 obj
<<
  /Pages 2 0 R
>>
endobj

2 0 obj
<<
  /Type /Pages
  /Count 1
  /Kids [3 0 R]
>>
endobj

3 0 obj
<<
  /Type /Page
  /Contents 4 0 R
  /Parent 2 0 R
  /Resources <<
    /Font <<
      /F1 <<
        /Type /Font
        /Subtype /Type1
        /BaseFont /Arial
      >>
<< /Length 47 >>
stream
BT
  /F1 110
  Tf
  10 400 Td
  (Hello World!)Tj
ET
endstream
endobj

xref
0 5
0000000000 65535 f
0000000010 00000 n
0000000047 00000 n
0000000111 00000 n
0000000313 00000 n

trailer
<<
  /Root 1 0 R
>>

startxref
416
%%EOF
```

HEADER

```
%PDF-1.1
```

BODY

```
010:
    1 0 obj
    <<
      /Pages 2 0 R
    >>
    endobj
047:
    2 0 obj
    <<
      /Type /Pages
      /Count 1
      /Kids [3 0 R]
    >>
    endobj
111:
    3 0 obj
    <<
      /Type /Page
      /Contents 4 0 R
      /Parent 2 0 R
      /Resources <<
        /Font <<
          /F1 <<
            /Type /Font
            /Subtype /Type1
            /BaseFont /Arial
          >>
        >>
      >>
    >>
    endobj
313:
    4 0 obj
    << /Length 47 >>
    stream
    BT                         BEGIN TEXT
      /F1 110                    FONT F1 (ARIAL) SET TO SIZE 110
      Tf                         SELECT THIS FONT
      10 400 Td                  MOVE TO COORDINATE 10, 400
      (Hello World!)Tj           OUTPUT TEXT "HELLO WORLD!"
    ET                         END TEXT
    endstream
    endobj
```

XREF (CROSS REFERENCE) TABLE

```
416:
    xref                       CROSS REFERENCES
    0 5                        5 OBJECTS, STARTING AT INDEX 0
    0000000000 65535 f         (STANDARD FIRST EMPTY OBJECT 0
    0000000010 00000 n         OFFSET TO OBJECT 1, REV 0
    0000000047 00000 n         TO OBJECT 2...
    0000000111 00000 n         3...
    0000000313 00000 n         4
```

TRAILER

```
    trailer
    <<
      /Root 1 0 R
    >>

    startxref
    416
    %%EOF
```

## BASICS

PDF IS TEXT BASED, WITH BINARY STREAMS

### TYPES

(): STRING
  EX: `(Hello World!)`
/NAME (IDENTIFIERS)
  EX: `/Count 1`
<<>>: DICTIONARY
  EX: `<</key1 value1 /key2 value2>>`
[]: ARRAY
  EX: `[0 1 2 3 4]`

### OBJECT REFERENCES

CONTENT IS STORED IN OBJECT
MOST CONTENT CAN BE INLINED OR REFERENCED IN A SEPARATE OBJECT

`/Key1 value` IS EQUIVALENT TO `/Key1 3 0 R` <OBJECT NUMBER> <REVISION NUMBER> R

```
[...]

3 0 obj
value
endobj
```

### BINARY STREAMS

BINARY STREAM ARE STORED IN SEPARATE OBJECTS LIKE THIS:

```
<object number> <object revision> obj
<< <STREAM METADATA> >>
stream
<STREAM CONTENT>
endstream
endobj
```

STREAM LENGTH, COMPRESSION PARAMETERS....

## TRIVIA

THE PDF WAS FIRST SPECIFIED BY ADOBE SYSTEMS IN 1993

INITIAL VERSIONS OF ADOBE ACROBAT WERE NOT FREE

## FILE STRUCTURE

### HEAD OF THE FILE

THE "%PDF-" SIGNATURE IDENTIFIES THE FORMAT
AND REQUIRED VERSION

### XREF

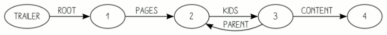
`xref`
<STARTING OBJECT> <OBJECT COUNT>
FOLLOWED BY XREF ENTRIES:
IF (OBJECT IN USE)
  <OFFSET:10> <GENERATION:5> `n`
ELSE
  <NEXT_FREE_OBJECT:10> <GENERATION:5> `f`

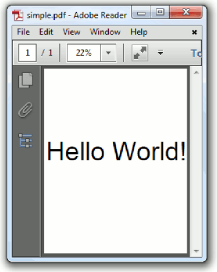
### END OF THE FILE

`startxref`
<XREF OFFSET IN DECODED STREAM>
`%%EOF`

## PARSING

THE HEADER `%PDF-1.?` SIGNATURE IS CHECKED TO IDENTIFY THE FILE FORMAT
THE XREF IS LOCATED VIA THE `startxref` OFFSET
THE `xref` TABLE GIVES OFFSET OF EACH OBJECT
THE `trailer` IS PARSED
EACH OBJECT REFERENCE IS FOLLOWED, BUILDING THE DOCUMENT
PAGES ARE CREATED, TEXT IS RENDERED





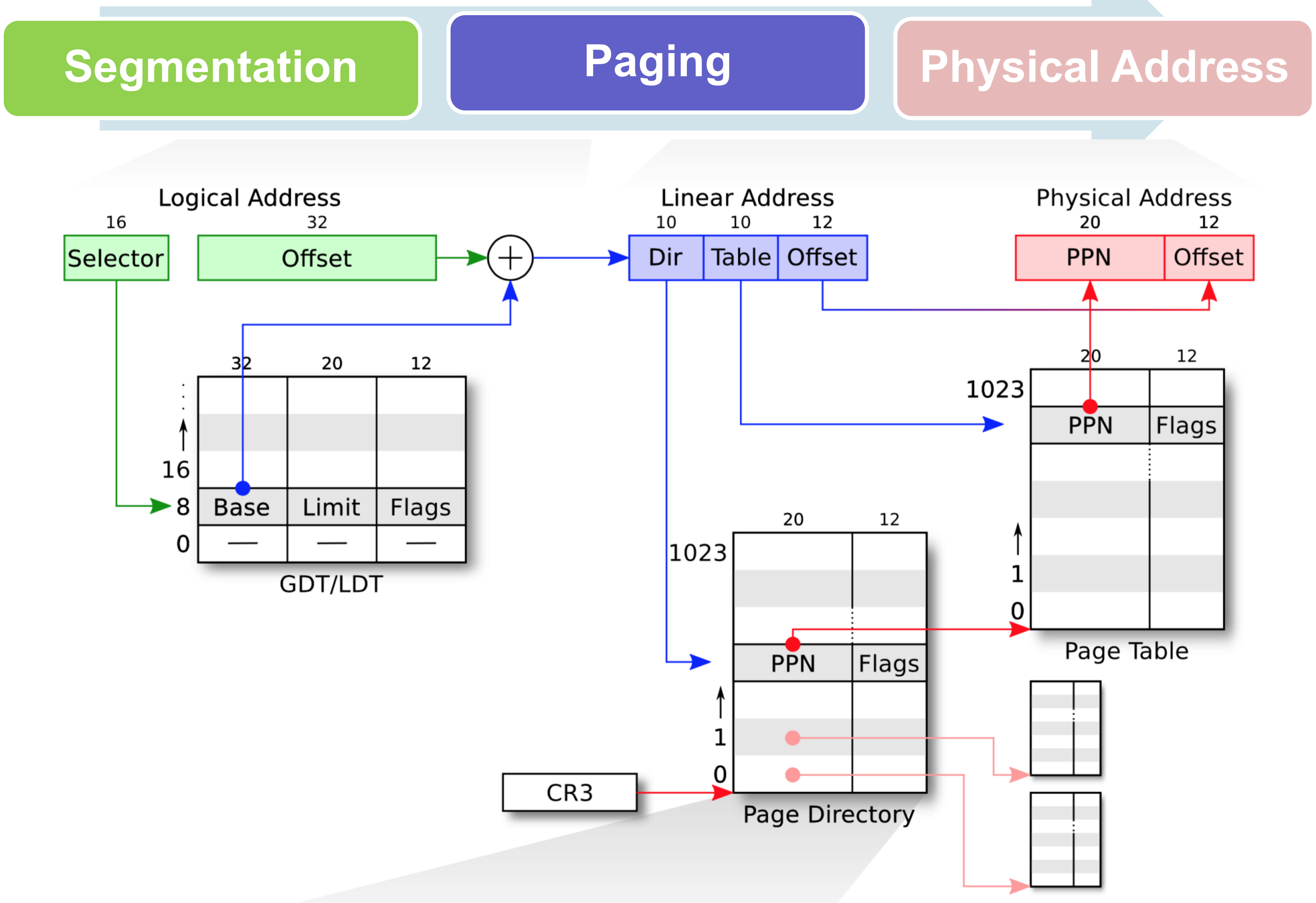
# PDF File Format

# MEMORY ANALYSIS…

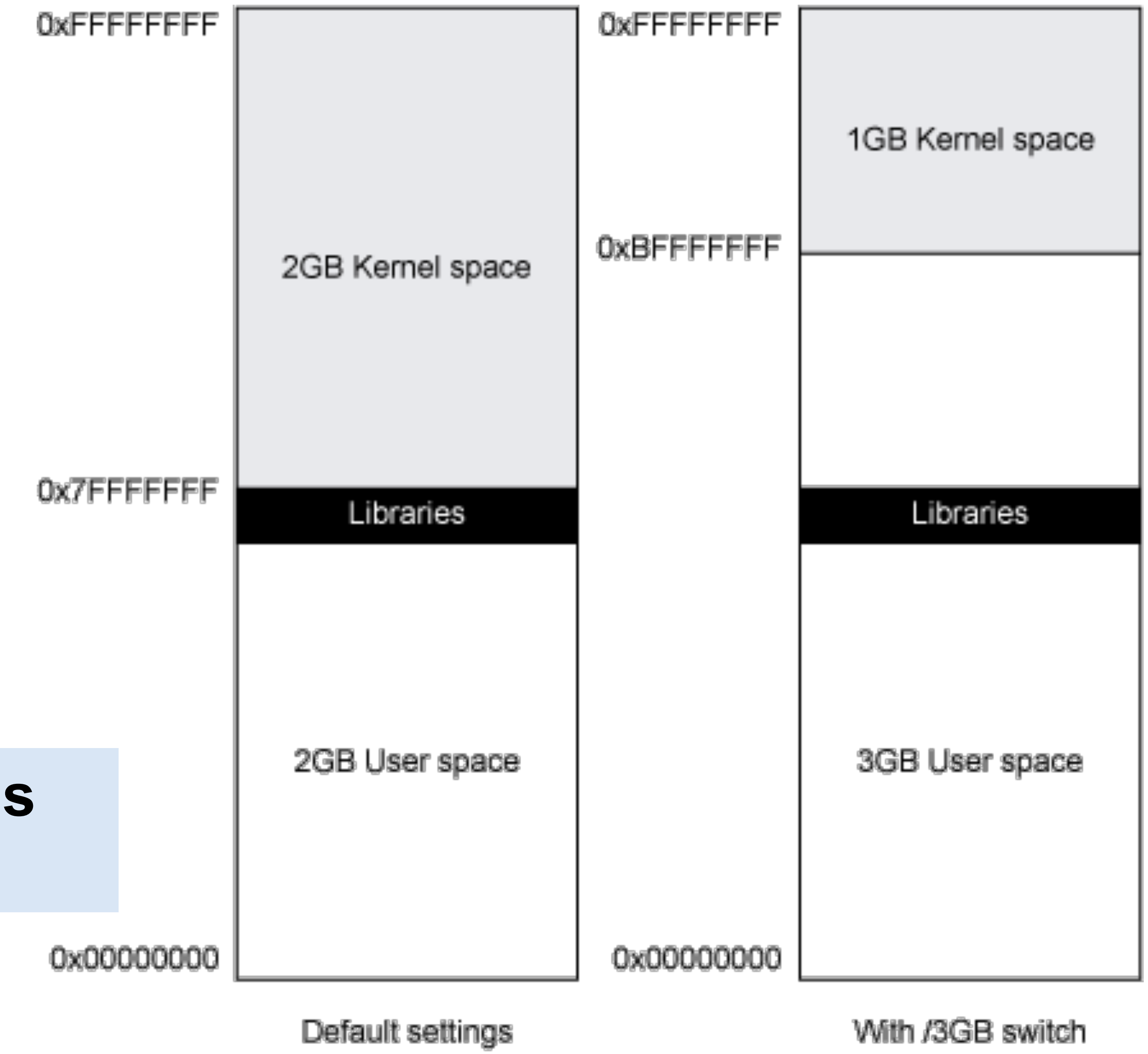
‘cause reverse engineering ninjas are busy

# x86/x64 Memory organization

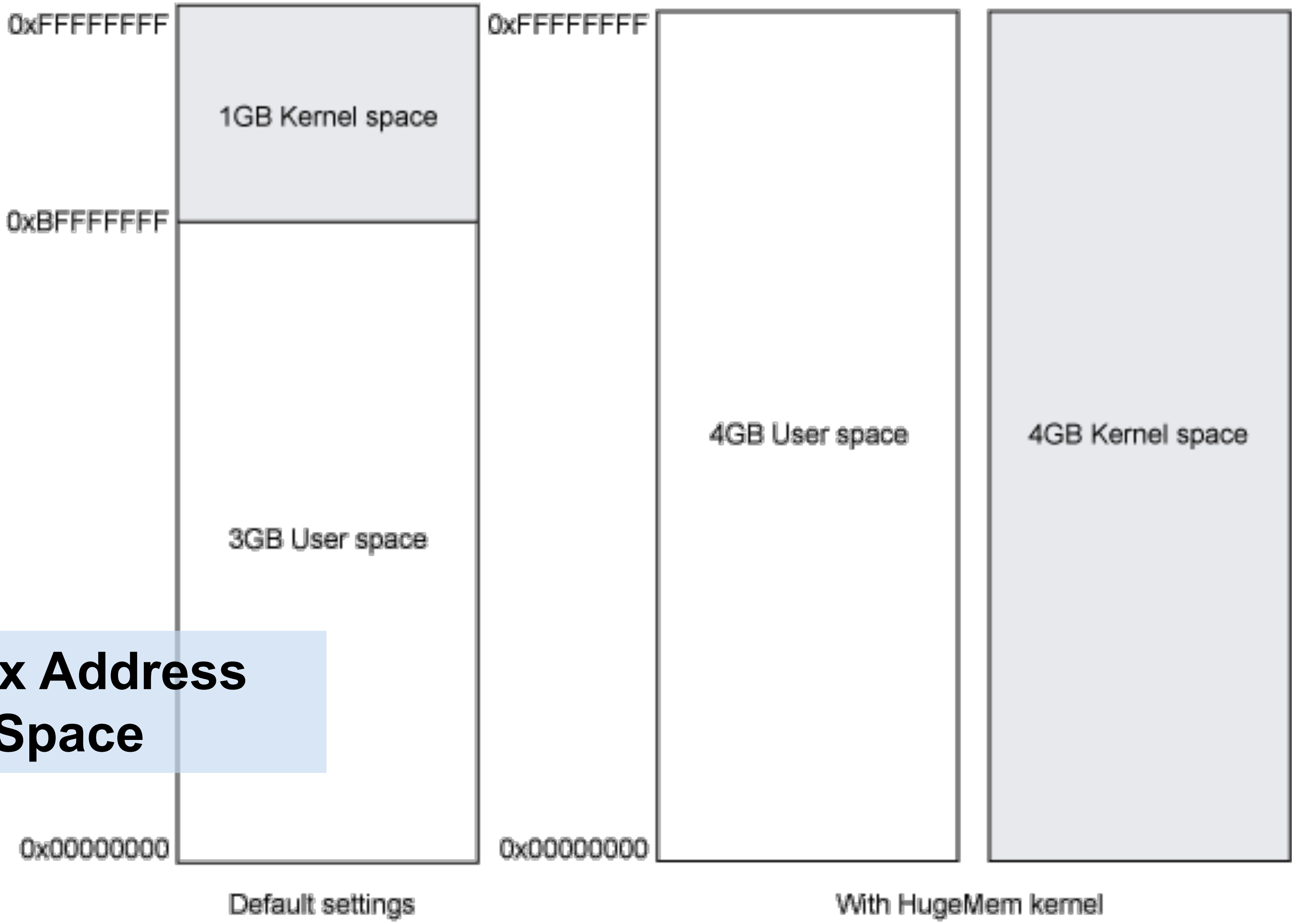
- Physical memory
  - RAM; what we really have installed
- Virtual memory
  - Separation of logical process memory from the physical
  - Logical address space > physical (e.g. swap)
  - Address space shared by several processes, yet separated
- Paging vs. Segmentation
  - Possible memory organization approaches


Segmentation
Paging
Physical Address
Logical Address
16
32
Selector
Offset
Linear Address
10
10
12
Dir
Table
Offset
Physical Address
20
12
PPN
Offset
32
20
12
16
8
0
Base
Limit
Flags
GDT/LDT
20
12
1023
PPN
Flags
1
0
Page Table
20
12
1023
PPN
Flags
1
0
Page Directory
CR3

# Win32 Address Space



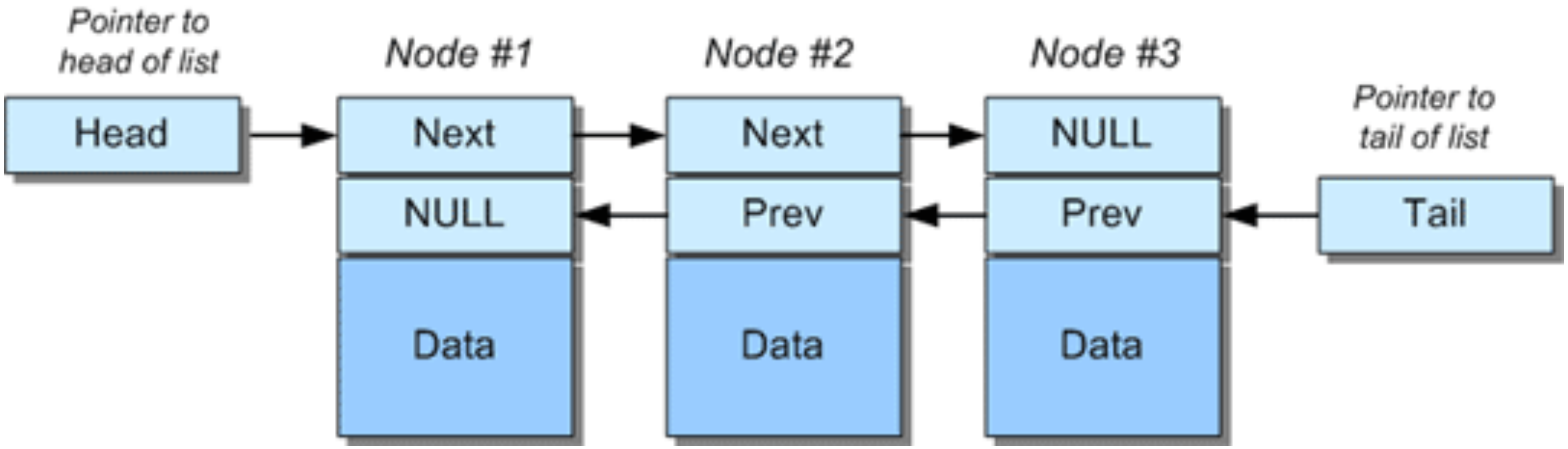
# Linux Address Space



# Operating System Data Structures

- How the OS knows about processes, files, …?
  - A lot of ‘metadata’ for important data
  - Based on C/C++ data structures (see MSDN documentation)
- (Double-)linked list
  - Another common data structure (not only in OS)
  - Method for implementing lists in computer memory
- Direct Kernel Object Manipulation (DKOM)
  - Used for manipulating the structures to hide malicious stuff

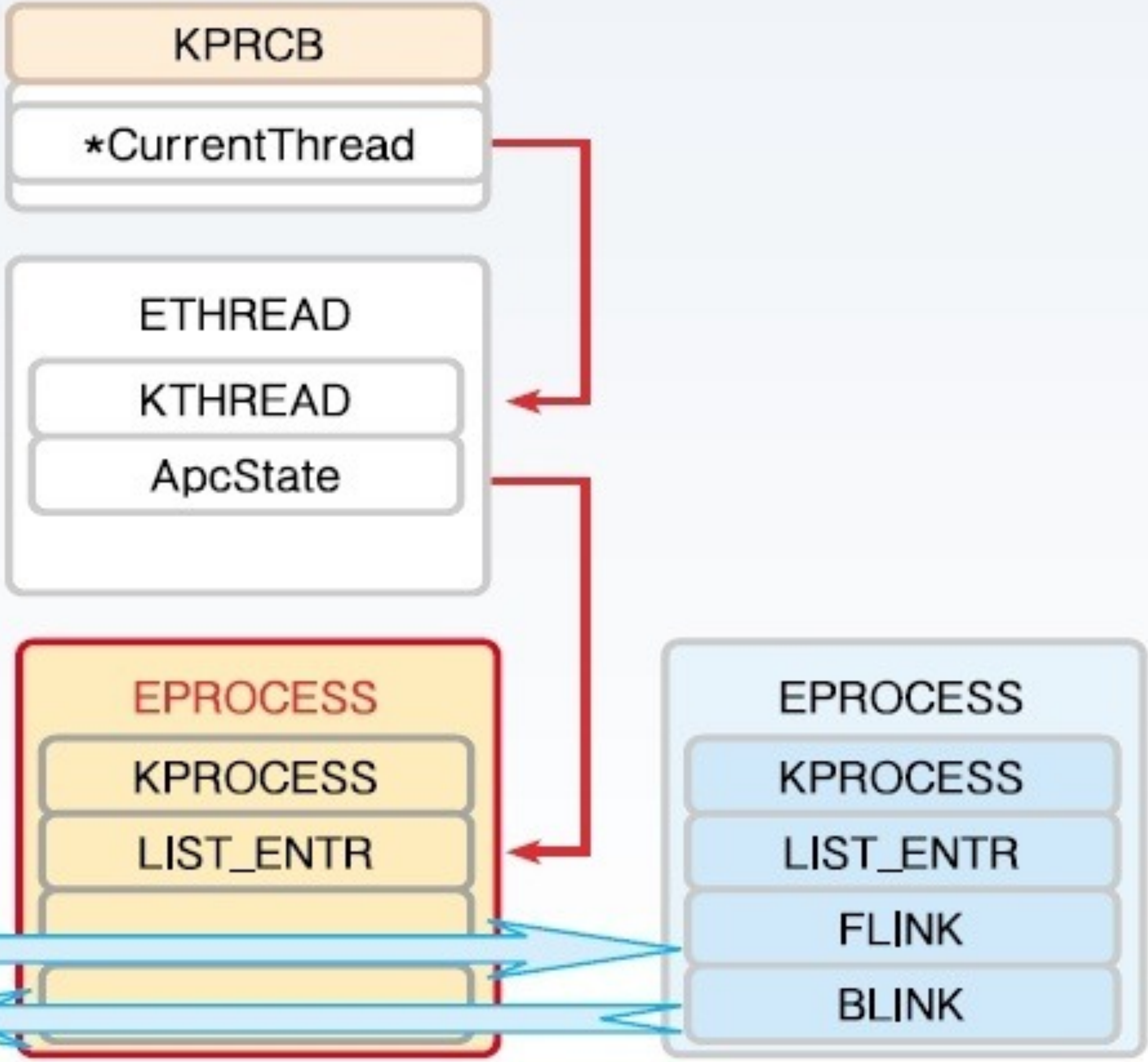
# Double Linked Lists



# DKOM – Direct Kernel Object Manipulation

- Dozens of various (double-)linked lists in Win32
  - Maintained by kernel
  - Processes, threads, opened files, memory allocations, …
- DKOM is used by rootkits
  - Hiding from the sight of the user
- Rootkit paradox
  - Rootkits need to run on the system
  - … and need to remain hidden at the same time
- Memory analysis can help to discover DKOM
  - Anti-analysis techniques are known as well

# Windows Process Structures



# Interesting OS Structures

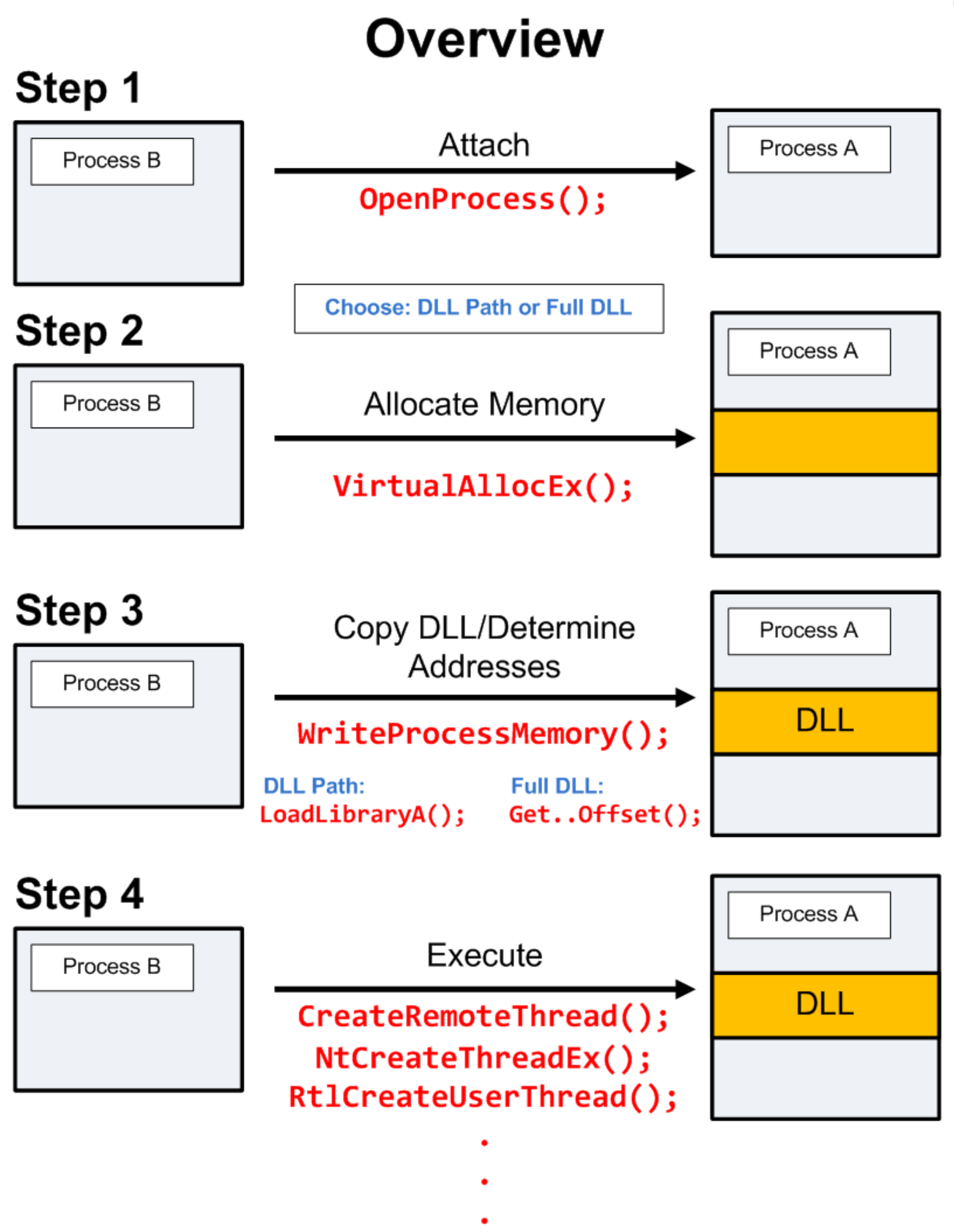
- Suspicious Memory Pages
- Processes
- Threads
- Sockets (Connections)
- Handles (Files)
- Modules/Libraries
- Mutexes
- LSA (Local Security Authority)
- Registry
- …

# Memory Pages

- Various ‘flags’
  - Read/write/executable pages
  - Helping OS to organize memory efficiently
- Executable + Writable pages
  - Why is it bad?
- **Process Injection technique**
  - Allocating a memory that can be modified (unpacked, decoded, decrypted) and executed.
  - Used by legitimate processes too (Windows OLE)

# DLL/Process Injection

So that Internet Explorer behaves like a malicious process…



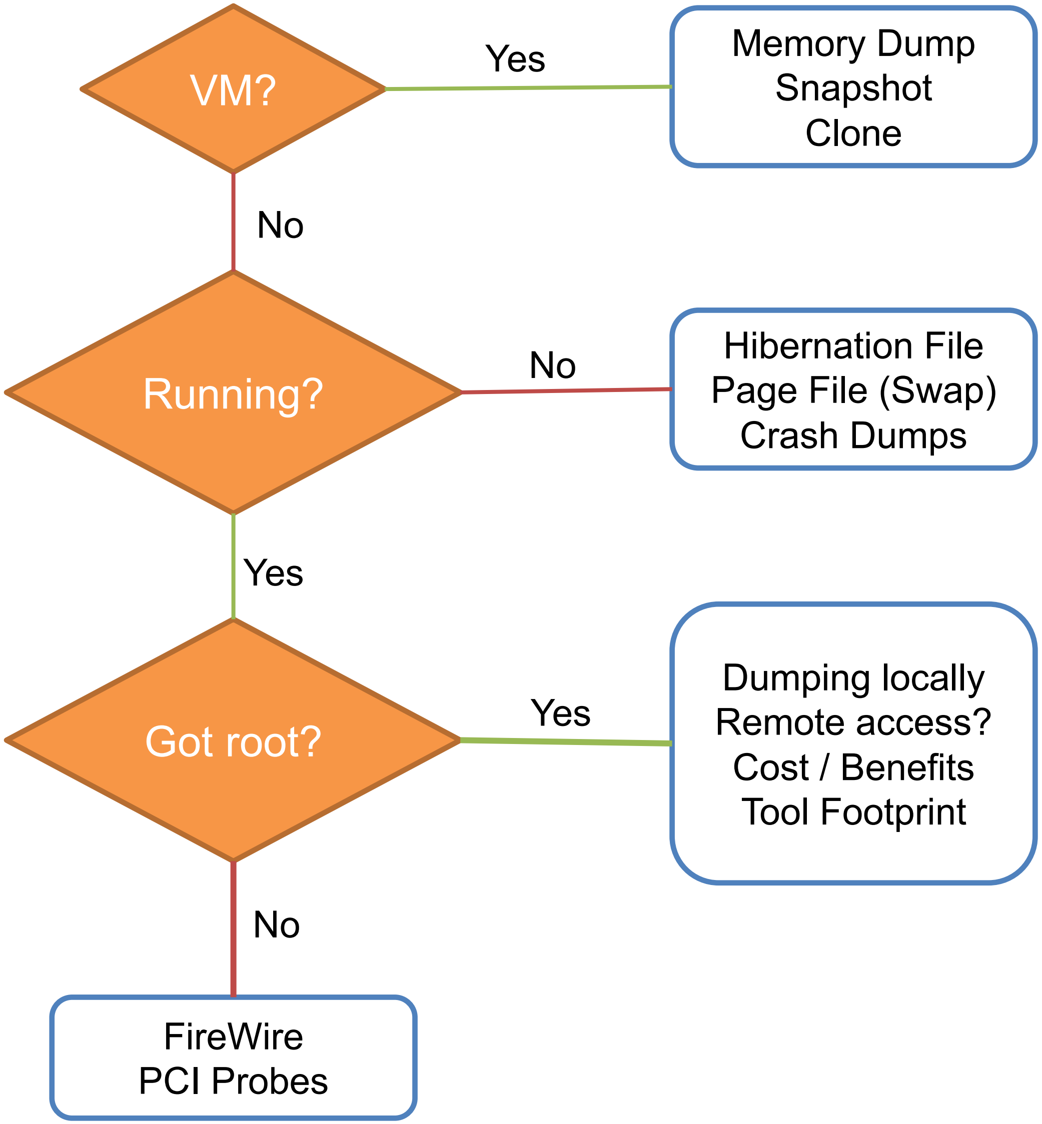
# And now something completely…

PRACTICAL

# Memory (re)sources

- Live RAM
  - – The most common source for analysis
  - – Easier to obtain from virtualized hosts
- Paging file/Swap
  - – Used by operating systems to allocate more memory then available RAM
- Hibernation file
- Memory crash dumps
  - – Very limited analysis options

# Memory Acquisition



# Memory Acquisition

- **Virtual Machines**
  - VMWare, VirtualBox, …
  - `VirtualBox -dbg -startvm “MalwareVM”` (and `.pgmphystofile` command)
- Directly from the system! (if we have system rights to do that)
  - `windd`, `fastdump`, `memoryze`
  - Or we can hibernate the system (hiberfil.sys)
- Remotely
  - Encase Enterprise, Mandiant Intelligent Response, Access Data FTK
- Common issues
  - Unsupported OS (Linux, MacOS; 32bit/64bit)
  - Swap (portions of memory on drive)
  - Malware not running inside a virtual machine

# Memory Acquisition (2)

- Local memory acquisition notes
  - Unless you have plenty of money, try to get root/admin access to the host
  - Better to acquire to external storage (USB, network)
  - The lower tool’s memory footprint, the better
  - If you run malware in VM, better have less RAM
    - Faster analysis
    - .. And configure no swap for the system too

# Memory Acquisition (3)

- Remote memory acquisition
  - Very useful for fast Incident Response
  - Requires enterprise licenses for the commercial tools
  - Acquisition is done over network
  - Agents already in memory, no extra memory demands
- Open source alternative?
  - GRR (Google Rapid Response)
  - Still in development, primarily Incident Response tool
  - Allows remote memory acquisition

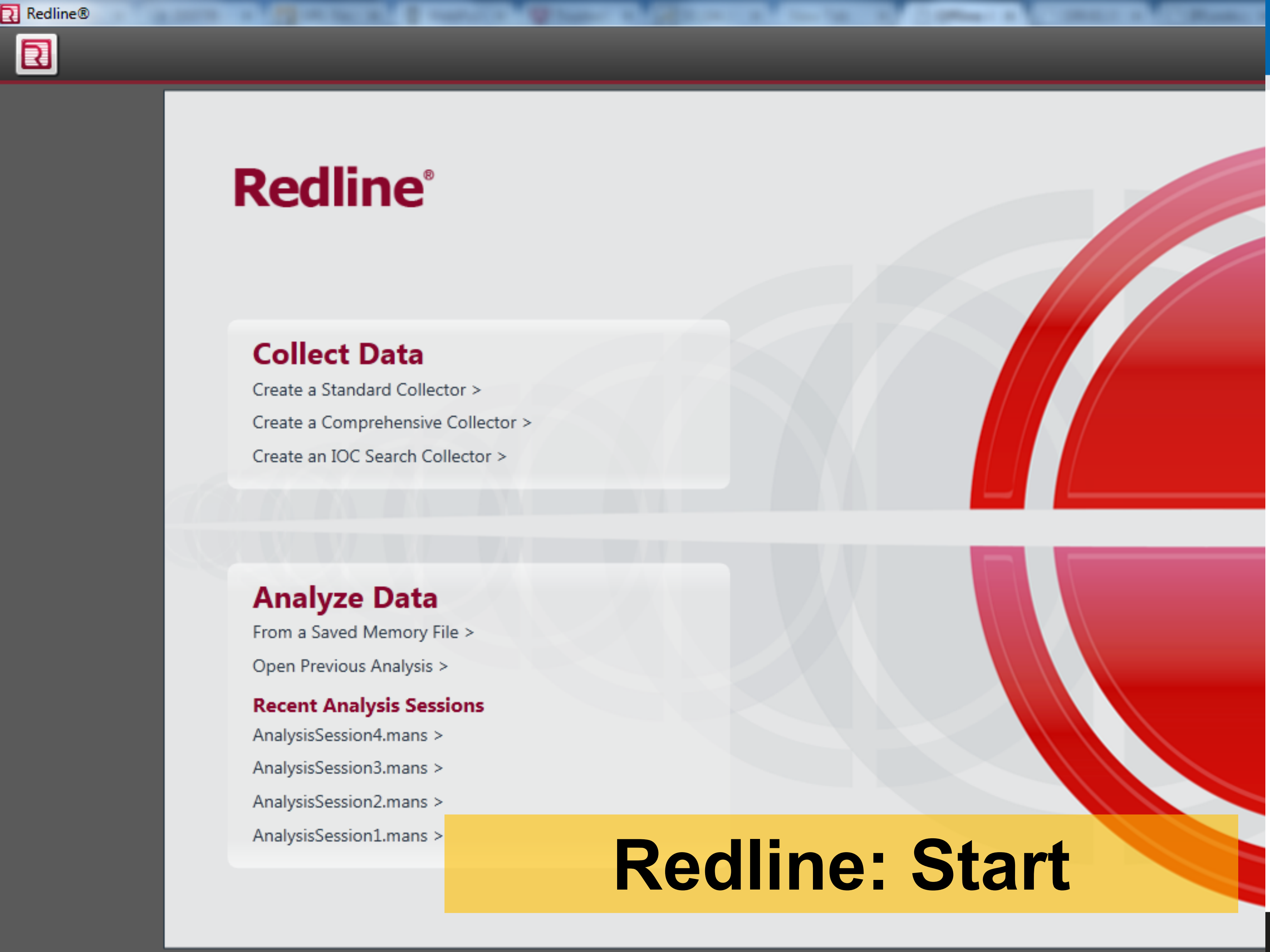
# Memory Analysis Tools

- Mandiant Redline
  - Free, available for Windows
- HBGary Responder (CE/Pro)
  - Community Edition available against registration
- Volatility Framework
  - Open source, no GUI
- Rekall
  - Open source, ‘Volatility done right’, GUI
  - Google supported (part of GRR agent)

# Mandiant/FireEye Redline

- Free tool for Incident Response
  - Not open-source, though
  - .NET executable (runs only under Windows)
- Nice and simple user interface
  - Very nice analysis workflow
  - Perfect for searching for string information
  - Rates the level of suspiciousness over processes
- Sad things
  - Memory analysis not reliable, process rating as well

# Redline: Start



Redline®

Redline®

## Collect Data

Create a Standard Collector >

Create a Comprehensive Collector >

Create an IOC Search Collector >

## Analyze Data

From a Saved Memory File >

Open Previous Analysis >

### Recent Analysis Sessions

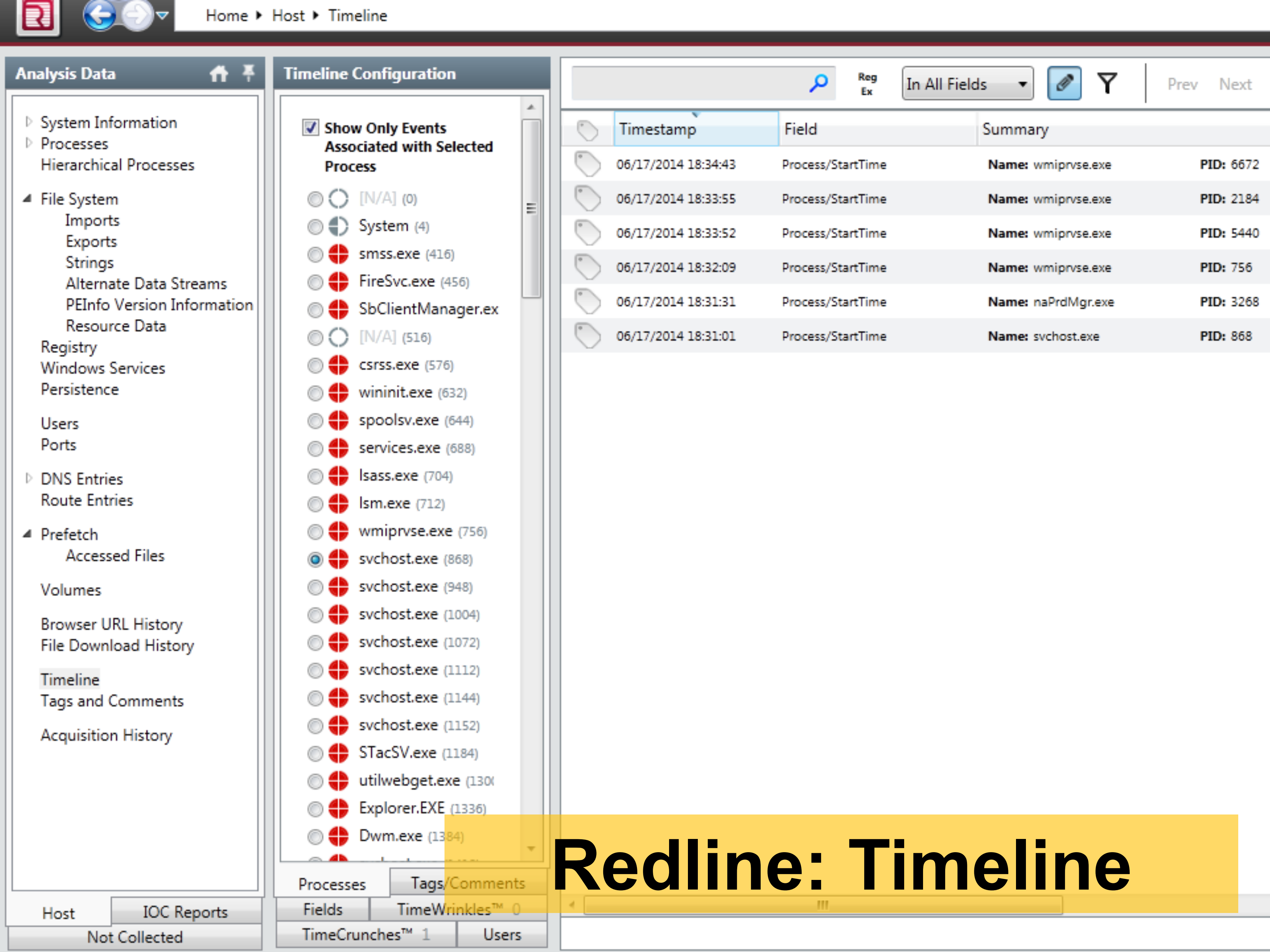
AnalysisSession4.mans >

AnalysisSession3.mans >

AnalysisSession2.mans >

AnalysisSession1.mans >

# Redline: Timeline



Home ▸ Host ▸ Timeline

**Analysis Data**

- System Information
- Processes
- Hierarchical Processes
- File System
  - Imports
  - Exports
  - Strings
  - Alternate Data Streams
  - PEInfo Version Information
  - Resource Data
- Registry
- Windows Services
- Persistence
- Users
- Ports
- DNS Entries
- Route Entries
- Prefetch
  - Accessed Files
- Volumes
- Browser URL History
- File Download History
- Timeline
- Tags and Comments
- Acquisition History

Host | IOC Reports | Not Collected

**Timeline Configuration**

☑ **Show Only Events Associated with Selected Process**

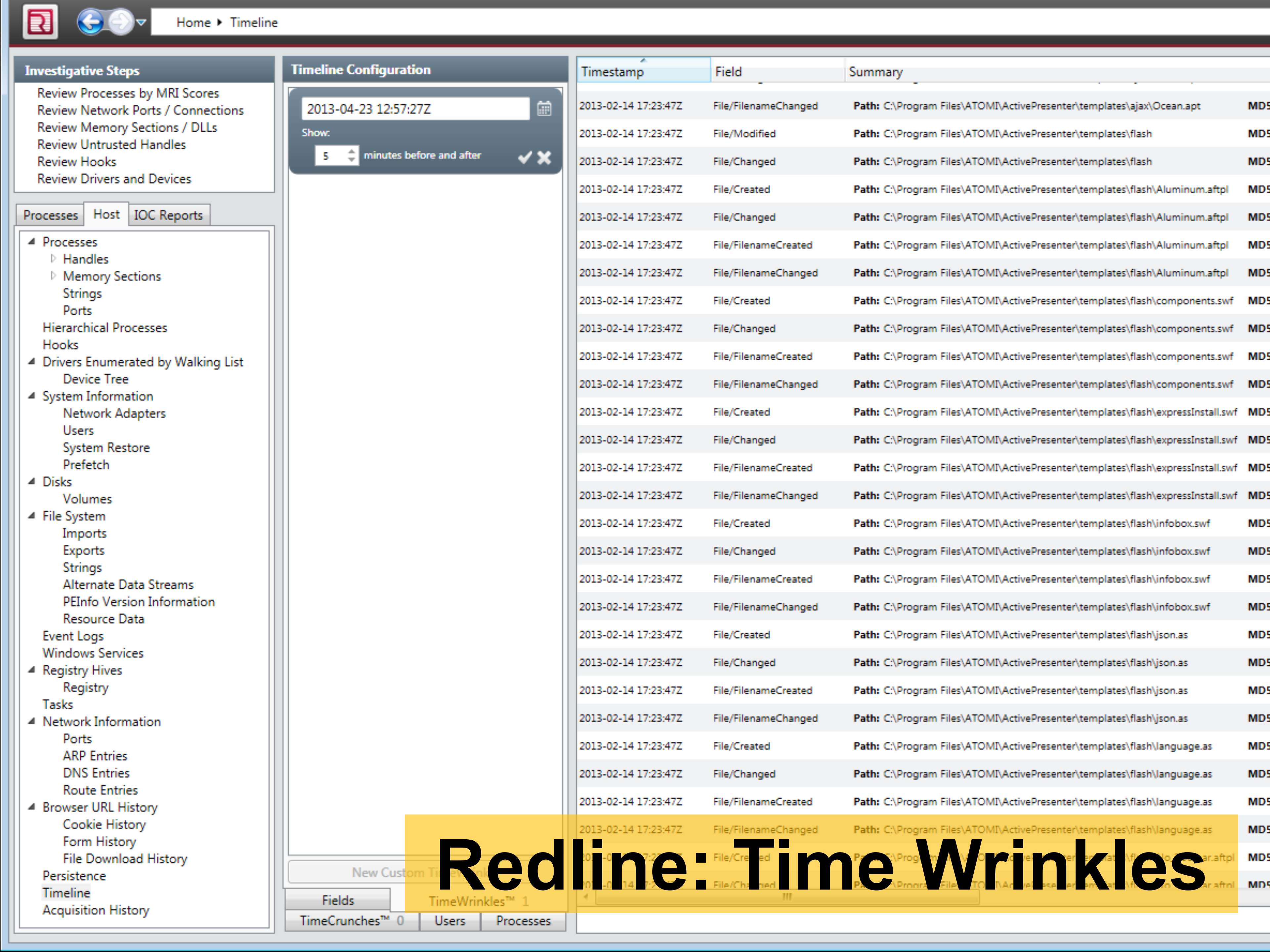
- ○ [N/A] (0)
- ○ System (4)
- ○ smss.exe (416)
- ○ FireSvc.exe (456)
- ○ SbClientManager.ex
- ○ [N/A] (516)
- ○ csrss.exe (576)
- ○ wininit.exe (632)
- ○ spoolsv.exe (644)
- ○ services.exe (688)
- ○ lsass.exe (704)
- ○ lsm.exe (712)
- ○ wmiprvse.exe (756)
- ● svchost.exe (868)
- ○ svchost.exe (948)
- ○ svchost.exe (1004)
- ○ svchost.exe (1072)
- ○ svchost.exe (1112)
- ○ svchost.exe (1144)
- ○ svchost.exe (1152)
- ○ STacSV.exe (1184)
- ○ utilwebget.exe (130
- ○ Explorer.EXE (1336)
- ○ Dwm.exe (1384)

Processes | Tags/Comments | Fields | TimeWrinkles™ 0 | TimeCrunches™ 1 | Users

Reg Ex | In All Fields | Prev | Next

| Timestamp | Field | Summary | |
|---|---|---|---|
| 06/17/2014 18:34:43 | Process/StartTime | **Name:** wmiprvse.exe | **PID:** 6672 |
| 06/17/2014 18:33:55 | Process/StartTime | **Name:** wmiprvse.exe | **PID:** 2184 |
| 06/17/2014 18:33:52 | Process/StartTime | **Name:** wmiprvse.exe | **PID:** 5440 |
| 06/17/2014 18:32:09 | Process/StartTime | **Name:** wmiprvse.exe | **PID:** 756 |
| 06/17/2014 18:31:31 | Process/StartTime | **Name:** naPrdMgr.exe | **PID:** 3268 |
| 06/17/2014 18:31:01 | Process/StartTime | **Name:** svchost.exe | **PID:** 868 |

# Redline: Time Wrinkles



2013-04-23 12:57:27Z

Show: 5 minutes before and after

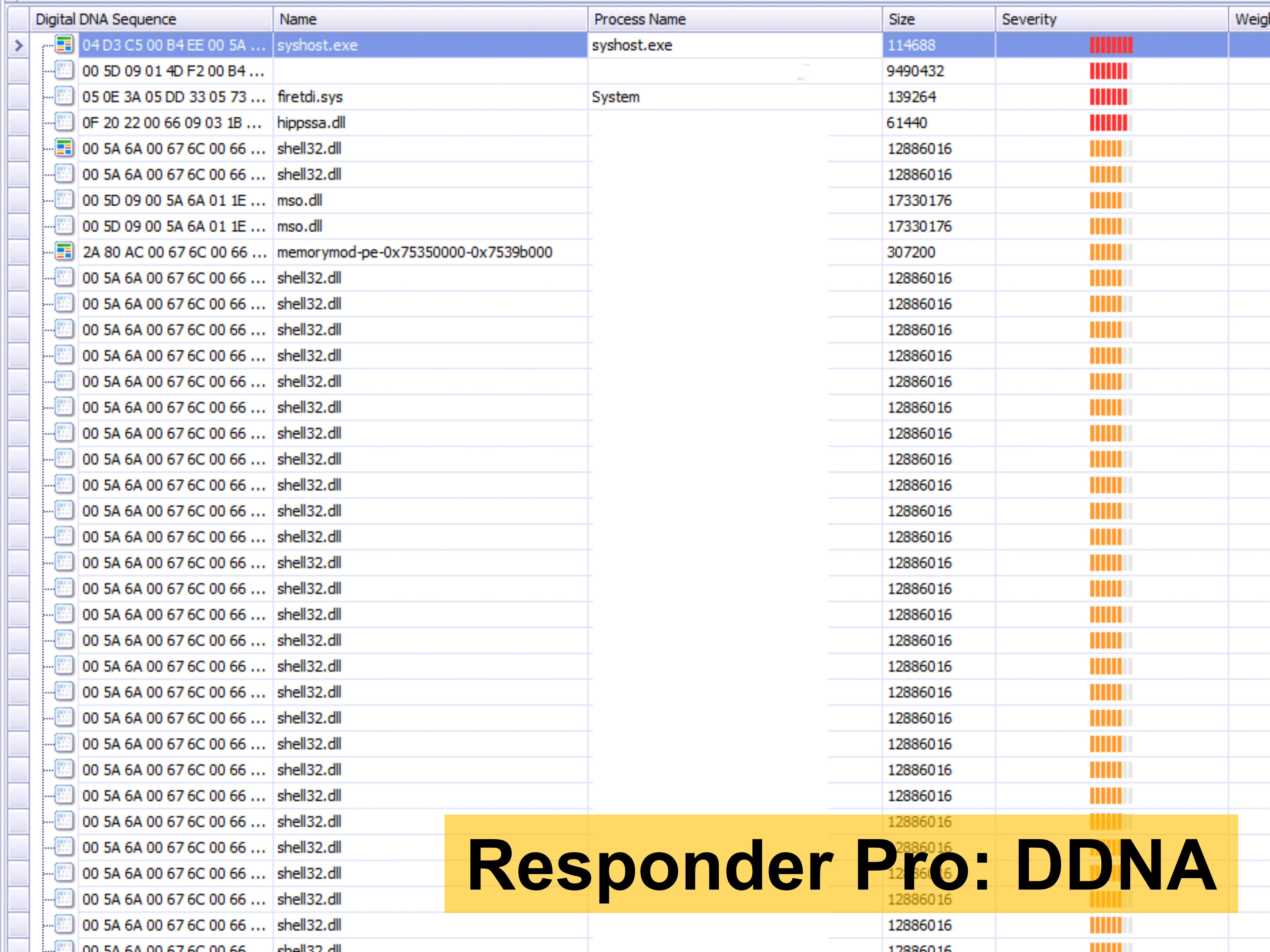
| Timestamp | Field | Summary | |
|---|---|---|---|
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\ajax\Ocean.apt | MD5 |
| 2013-02-14 17:23:47Z | File/Modified | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as | MD5 |
| 2013-02-14 17:23:47Z | File/Created | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as | MD5 |
| 2013-02-14 17:23:47Z | File/Changed | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameCreated | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as | MD5 |
| 2013-02-14 17:23:47Z | File/FilenameChanged | **Path:** C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as | MD5 |
| [illegible] | File/Created | [illegible] | MD5 |
| [illegible] | File/Changed | [illegible] | MD5 |

# HBGary Responder (Pro/CE)

- Professional Tool
  - Very expensive
  - Yet not very well maintained in the last few years
- Windows only
  - .NET written, supports only Windows images
- ‘Killer’ features
  - Digital DNA
    - automatic rating of suspicious processes
  - Visual ‘Canvas’ debugger
- Supports the analysis of (unpacked) binaries

# HBGary Responder Pro -- DDNA

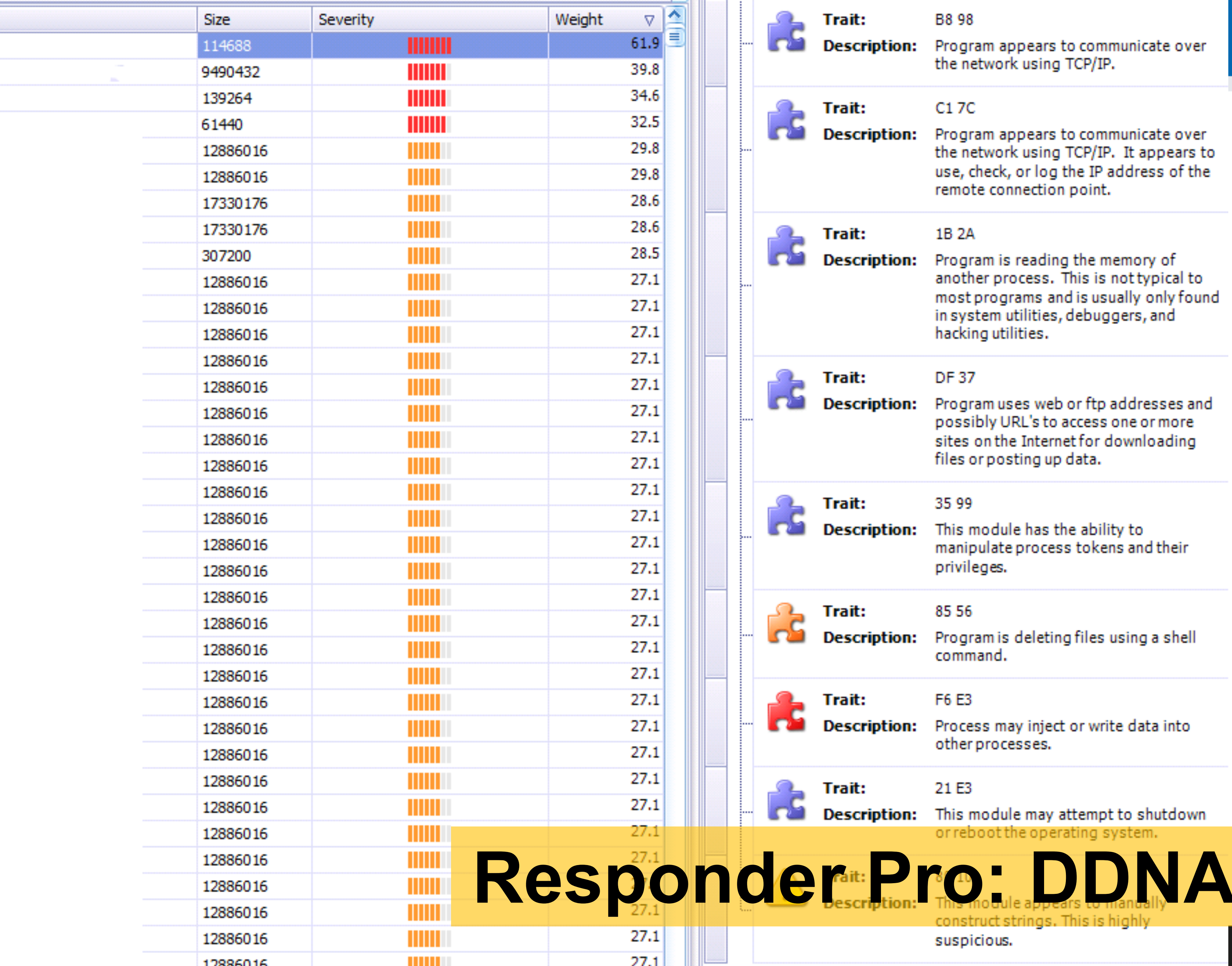
- Examples of the ‘reasoning’ behind DDNA
  - Does the process communicate over TCP/IP?
  - Does it manipulate with registry?
  - Did the analysis reveal any known bad stuff (strings, IPs, mutexes?)
  - Does the process access any other process in the system?
  - Does it access some system-critical process?
  - Did the analysis find any evidence of obfuscation?
  - …



# Responder Pro: DDNA

| Digital DNA Sequence | Name | Process Name | Size | Severity | Weig |
|---|---|---|---|---|---|
| 04 D3 C5 00 B4 EE 00 5A ... | syshost.exe | syshost.exe | 114688 | 🟥 | |
| 00 5D 09 01 4D F2 00 B4 ... | | | 9490432 | 🟥 | |
| 05 0E 3A 05 DD 33 05 73 ... | firetdi.sys | System | 139264 | 🟥 | |
| 0F 20 22 00 66 09 03 1B ... | hippssa.dll | | 61440 | 🟥 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5D 09 00 5A 6A 01 1E ... | mso.dll | | 17330176 | 🟧 | |
| 00 5D 09 00 5A 6A 01 1E ... | mso.dll | | 17330176 | 🟧 | |
| 2A 80 AC 00 67 6C 00 66 ... | memorymod-pe-0x75350000-0x7539b000 | | 307200 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |
| 00 5A 6A 00 67 6C 00 66 ... | shell32.dll | | 12886016 | 🟧 | |

# Responder Pro: DDNA



| Size | Severity | Weight |
|---|---|---|
| 114688 | 🟥 | 61.9 |
| 9490432 | 🟥 | 39.8 |
| 139264 | 🟥 | 34.6 |
| 61440 | 🟥 | 32.5 |
| 12886016 | 🟧 | 29.8 |
| 12886016 | 🟧 | 29.8 |
| 17330176 | 🟧 | 28.6 |
| 17330176 | 🟧 | 28.6 |
| 307200 | 🟧 | 28.5 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |
| 12886016 | 🟧 | 27.1 |

**Trait:** B8 98
**Description:** Program appears to communicate over the network using TCP/IP.

**Trait:** C1 7C
**Description:** Program appears to communicate over the network using TCP/IP. It appears to use, check, or log the IP address of the remote connection point.

**Trait:** 1B 2A
**Description:** Program is reading the memory of another process. This is not typical to most programs and is usually only found in system utilities, debuggers, and hacking utilities.

**Trait:** DF 37
**Description:** Program uses web or ftp addresses and possibly URL's to access one or more sites on the Internet for downloading files or posting up data.

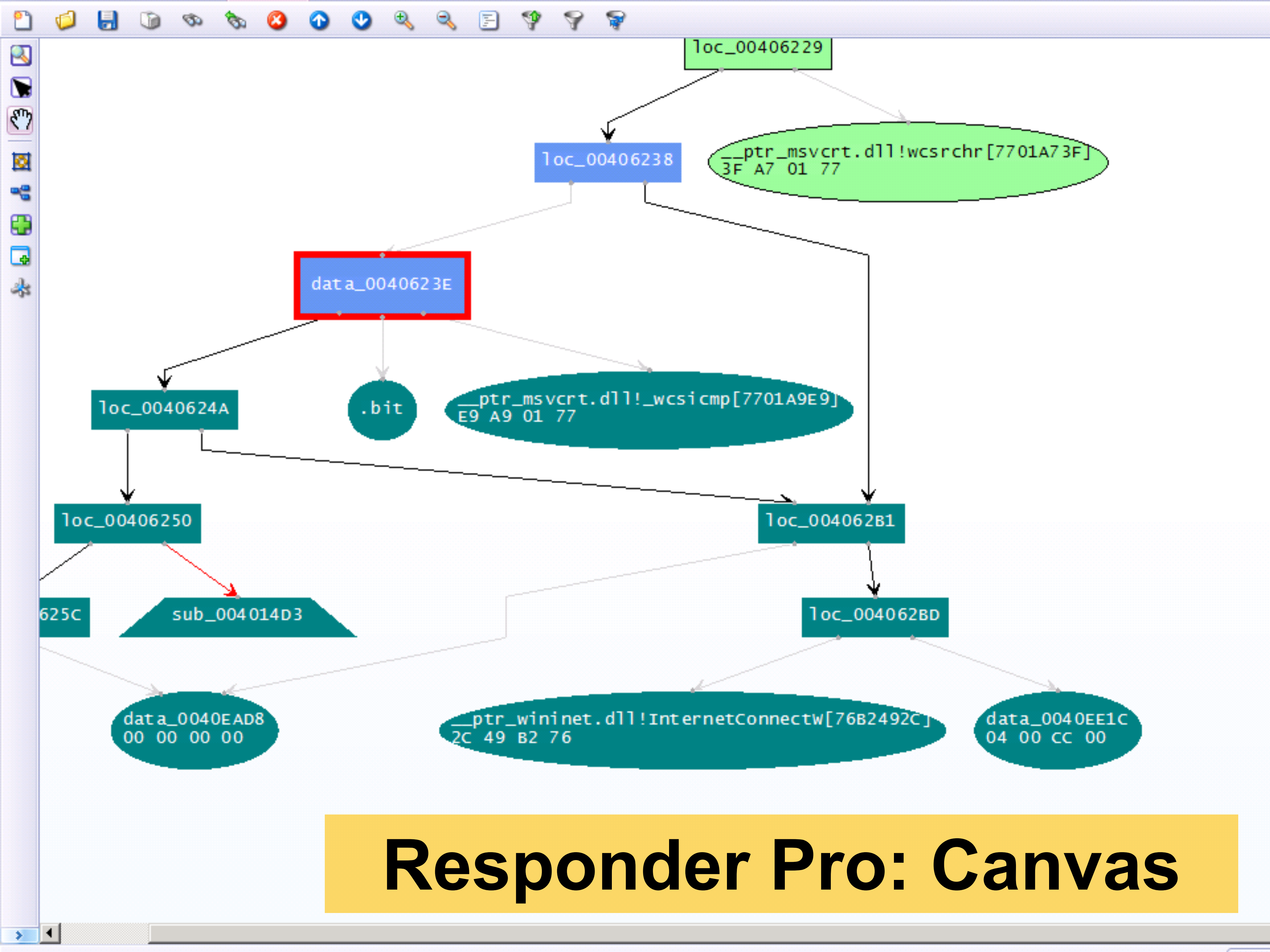
**Trait:** 35 99
**Description:** This module has the ability to manipulate process tokens and their privileges.

**Trait:** 85 56
**Description:** Program is deleting files using a shell command.

**Trait:** F6 E3
**Description:** Process may inject or write data into other processes.

**Trait:** 21 E3
**Description:** This module may attempt to shutdown or reboot the operating system.

**Trait:** [illegible]
**Description:** This module appears to manually construct strings. This is highly suspicious.



# Responder Pro: Canvas

# Volatility Framework

- Open source tool
  - GPL licensed
- Written in Python
  - Available for variety of platforms (Linux, Windows, Mac OS)
  - Can be automated; many contributed plugins
- Supports analysis of memory dumps from various OSs
  - Windows, Linux, MacOS, Android
  - Both 32-bit and 64-bit versions
- Command-line driven
- Two (experimental) web GUIs

# Google Rekall

- Another open source tool
- Supported by Google
  - Included as a part of GRR (Google Rapid Response) agent
- Originally based on the code of Volatility
  - Shared commands
  - Different architectural concepts
- Proof-of-concept GUI
  - Better workflows

# Additional Important Tools

- **Strings**
  - Both *nix and Windows
  - Extracts strings information from the file
  - Can be used in cooperation with Volatility/Rekall
  - Beware of text encoding! (ascii, utf-8, …)
- **Foremost**
  - Forensic tool
  - Can extract various data files from an image (or process)
    - Images, executables, documents, …

# Forensic analysis of RAM?

- Are there any benefits?
- Collecting forensic evidence
  - Executable images
  - PDF/Doc documents
    - Possible origin of the infection?
  - Images
  - URLs
- Getting approximate timeline
  - Works better on servers (always online, higher uptime, way more RAM)

# What to search for in Operating System?

- Command&Control (C2) communication
- Hidden processes
- Process/DLL injection evidence
- Non-standard/infamous binaries/mutexes
- Open sockets and files
- Registry records
- Command-line history
- Encryption keys!

# Known Bad Mutexes

- *Conficker*: `.*-7 and .*-99`
- *Sality.AA*: `Op1mutx9`
- *Flystud.??*: `Hacker.com.cn_MUTEX`
- *NetSky*: `'D'r'o'p'p'e'd'S'k'y'N'e't'`
- *Sality.W*: `u_joker_v3.06`
- *Poison Ivy*: `)!VoqA.I4` (and 10 thousand others)
- *Koobface*: `35fsdfsdfgfd5339`

# Known Good Processes/Locations

| Process Name | Expected Path |
|---|---|
| lsass.exe | \windows\system32 |
| services.exe | \windows\system32 |
| csrss.exe | \windows\system32 |
| explorer.exe | \windows |
| spoolsv.exe | \windows\system32 |
| smss.exe | \windows\system32 |
| svchost.exe | \windows\system32 |
| iexplore.exe | \program files<br>\program files (x86) |
| winlogon.exe | \windows\system32 |

# Operational Security (OpSec)

- Basics of OpSec
  - “Think before you act” mentality
  - Limited information sharing
- Specifics of memory analysis
  - You can often upload dumped executables to VirusTotal
    - md5 of the process is different from the executable
    - This doesn’t apply for documents/HTML pages!
  - **However, incomplete binaries still can infect your system!**
    - Running in VM or other OS is recommended

# Recommended Analysis Process

- **Use Internet!** (Google, VirusTotal, …)
- **Make notes!**
  - What OS is being analyzed? (imageinfo)
  - Network connections? (+ whois records, …)
  - Processes (hidden, odd, non-standard; timestamps, …)
  - Mutexes (+ files open)
  - Dump processes when needed (OpSec!)
  - Strings (URIs, C-like strings %s %d, domains, …)
- **Summarize your findings in final report**

# More information

- Web pages of this course
  - https://dior.ics.muni.cz/~valor/pv204/
- **Additional resources**
  - Public memory images for analysis
  - Reverse Engineering for Beginners (amazing PDF doc)
  - REMnux: All you need to start with RE
  - ContagioDump blog (for additional malware samples)

Thank you for your attention.

Answers & Questions

# LAB

# Lab Requirements

- Oracle VirtualBox
  - And enough space on your hard drive (12 GB at least)
- **Volatility Framework**
- Mandiant Redline
- Unix tools
  - `strings`, `foremost`
- Your favorite text editor for notes
- Javascript/PDF analysis tools

# Recommended Analysis Process

- **Use Internet!** (Google, VirusTotal, …)
- **Make notes!**
  - What OS is being analyzed?
  - Network connections? (+ whois records, …)
  - Processes (hidden, odd, non-standard; timestamps, …)
  - Mutexes (+ files open)
  - Strings (URIs, C-like strings %s %d, domains, …)
  - …
- **Summarize your findings in final report**

# Volatility Framework – cheat sheet

- `psxview` (search for hidden processes)
- `apihooks`
- `driverscan`
- `ssdt` / `driverirp` / `idt`
- `connections` / `connscan` (WinXP, active network connections)
- `netscan` (Win7, opened network sockets and connections)
- `pslist` / `psscan` (process listing from WinAPI vs. EPROCESS blocks)
- `malfind` / `ldrmodules` (code injection + dump / DLL detection)
- `hivelist` (registry lookup and parsing) / `hashdump`
- `handles` / `dlllist` / `filescan` (filelist / DLL files / FILE_OBJECT handles)
- `cmdscan` / `consoles` (`cmd.exe` history / console buffer)
- `shimcache` (application compatibility info)
- `memdump` / `procmemdump` / `procexedump`

# Analysis: xp-infected.vmem

- Recommended tools
  - Volatility, Rekall (or Redline)
- Objectives:
  - Get familiar with memory of your first infected system

# Analysis: win7_x64.vmem

- Recommended tools
  - Volatility, Rekall (or Redline)
- Objectives:
  - Get familiar with memory of Win7 x64 system
  - Can you see any differences from the previous sample?

# Analysis: zeus.vmem

- Recommended tools
  - Volatility, Rekall
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what infections this

# Analysis: zeus2x4.vmem

- Recommended tools
  - Volatility, Rekall
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what infections this
  - Can you dump the virus configuration?

# Analysis: bob.vmem

- Recommended tools
  - Volatility, Rekall, Foremost, Strings
- Objectives:
  - Find suspicious network connections
  - Find process responsible for the network activity
  - Can you figure out what caused the infection?
  - Can you dump the initial source vector?
  - What known vulnerability (CVE) has been exploited?

# More information

- Web pages of this course
  - `https://dior.ics.muni.cz/~valor/pv204/`
- **Additional resources**
  - Public memory images for analysis
  - Reverse Engineering for Beginners (amazing PDF doc)
  - REMnux: All you need to start with RE
  - ContagioDump blog (for additional malware samples)

Thank you for your attention.

Answers & Questions