PV204 Security technologies

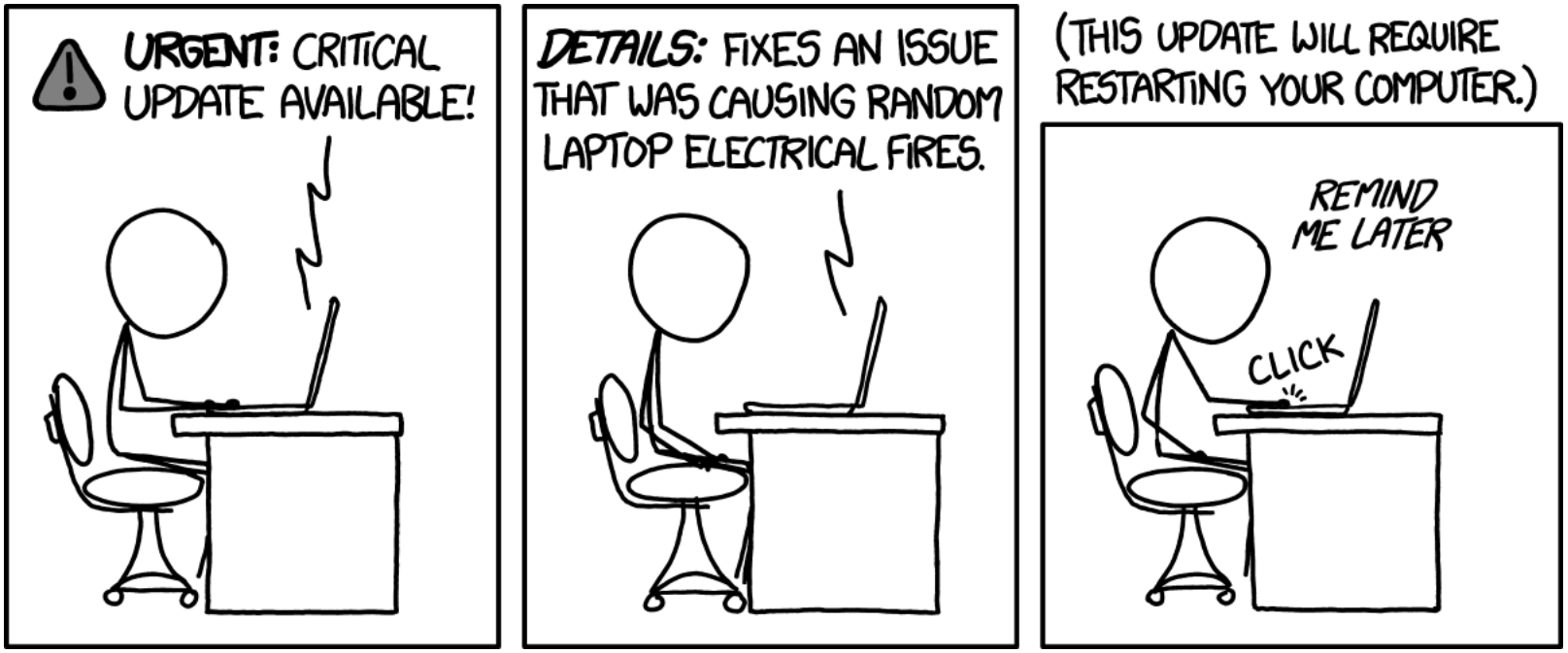
In-Memory Malware Analysis

Václav Lorenc

Security Operations Center Manager, Oracle + NetSuite

CROCS

Centre for Research on Cryptography and Security

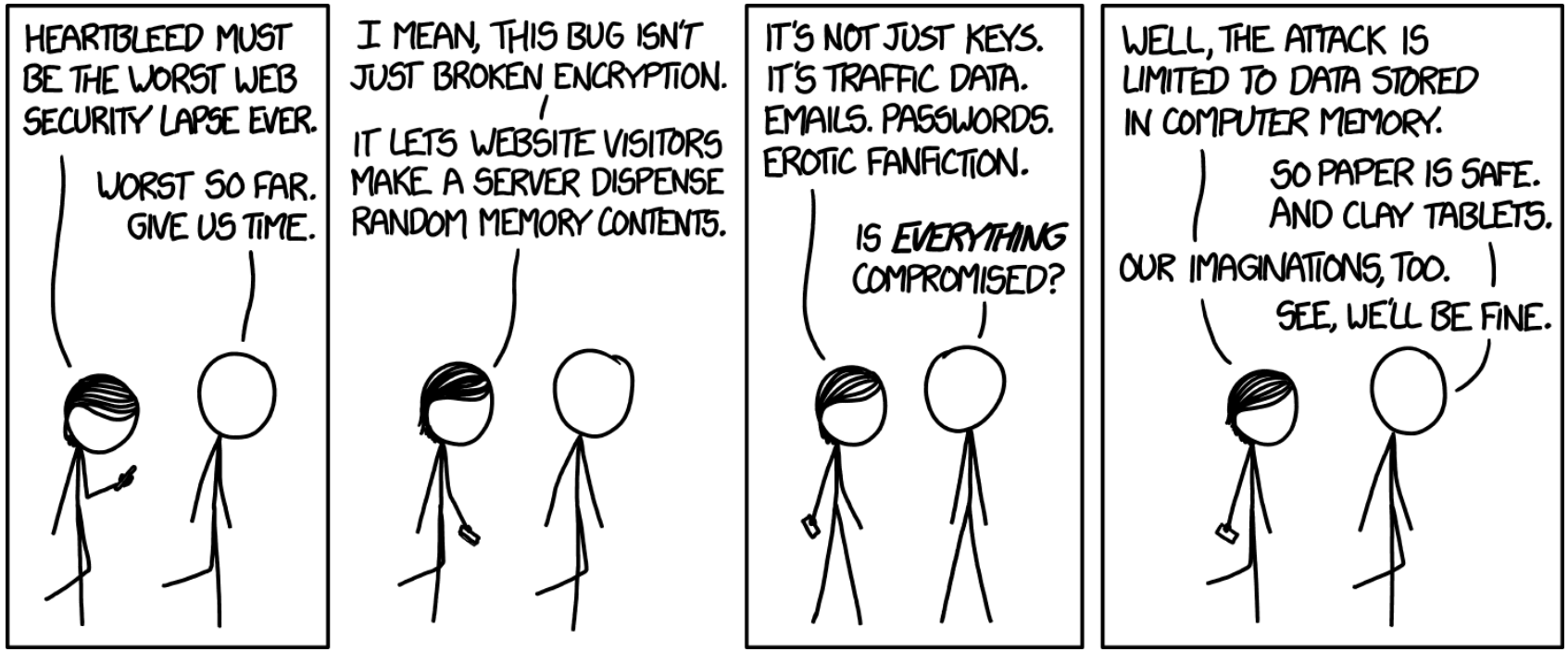

URGENT: CRITICAL UPDATE AVAILABLE!
DETAILS: FIXES AN ISSUE THAT WAS CAUSING RANDOM LAPTOP ELECTRICAL FIRES.
(THIS UPDATE WILL REQUIRE RESTARTING YOUR COMPUTER.)
REMIND ME LATER
CLICK

Agenda

- Basic intro
  - No x86 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?

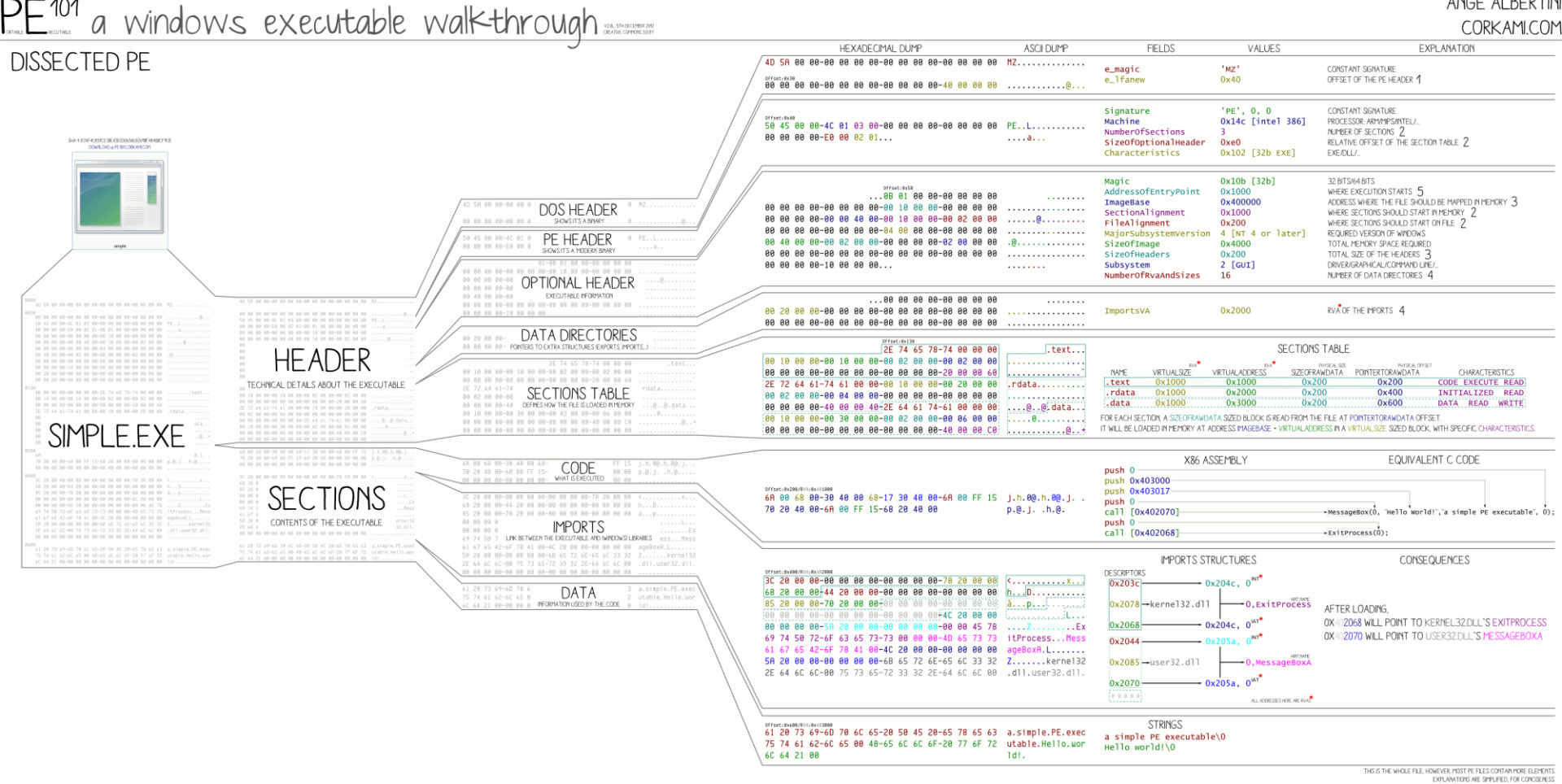
- **It's fun!**
- Acquiring evidence for legal investigations
  - It used to be different in the past
- Technical simplification of reverse engineering
  - No binary obfuscation present – the code has to run
- Incident response activities
  - Easy way how to learn more about the attackers
  - Malicious binary may only be present in memory
  - Fast: RAM is (usually) smaller than full hard-drive images


HEARTBLEED MUST BE THE WORST WEB SECURITY LAPSE EVER.
WORST SO FAR. GIVE US TIME.
I MEAN, THIS BUG ISN'T JUST BROKEN ENCRYPTION.
IT LETS WEBSITE VISITORS MAKE A SERVER DISPENSE RANDOM MEMORY CONTENTS.
IT'S NOT JUST KEYS. IT'S TRAFFIC DATA. EMAILS. PASSWORDS. EROTIC FANFICTION.
IS EVERYTHING COMPROMISED?
WELL, THE ATTACK IS LIMITED TO DATA STORED IN COMPUTER MEMORY.
SO PAPER IS SAFE. AND CLAY TABLETS.
OUR IMAGINATIONS, TOO.
SEE, WE'LL BE FINE.


HOTPLUG LT™
MOUSE JIGGLER
Prevents a computer from falling asleep while you work!
wiebeTECH
our best ideas store yours
PCD

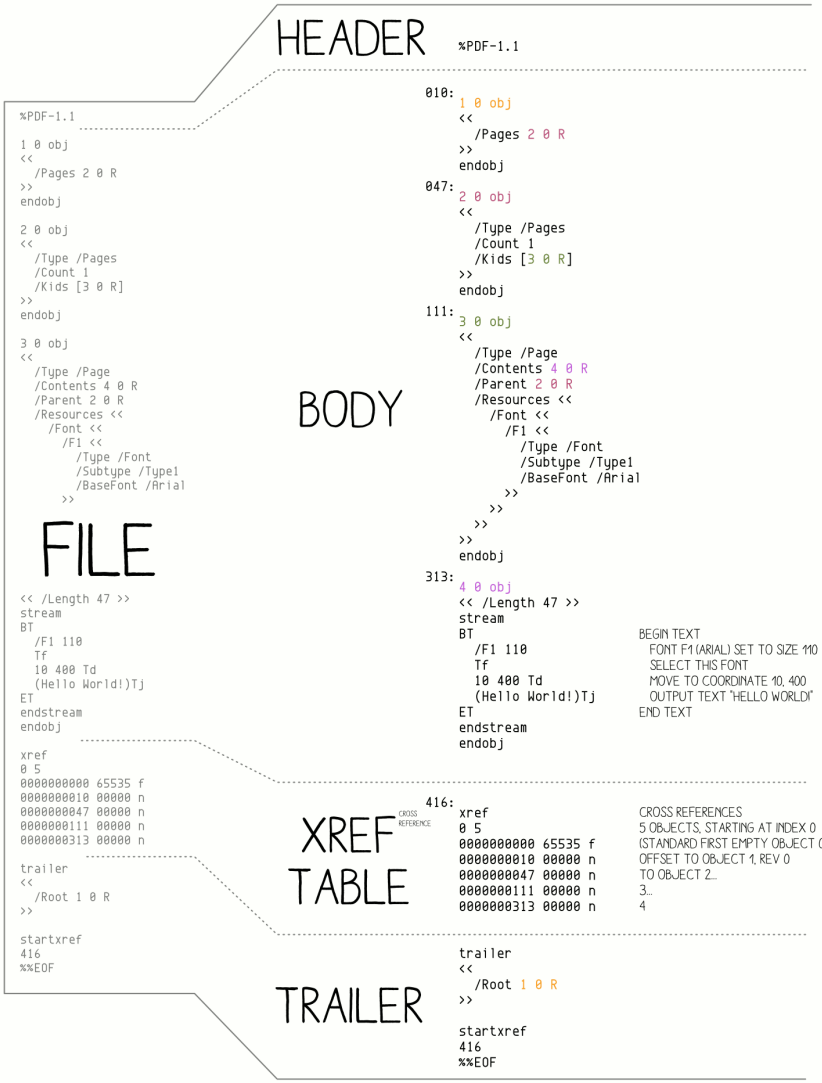
Challenges in Reverse Engineering (RE)

- 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 101
a windows executable walk-through
ANGE ALBERTINI
CORKAMI.COM
DISSECTED PE
HEXADECIMAL DUMP
ASCII DUMP
FIELDS
VALUES
EXPLANATION
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..)
HEADER
TECHNICAL DETAILS ABOUT THE EXECUTABLE
SIMPLE.EXE
CODE
WHAT IS EXECUTED
SECTIONS
CONTENTS OF THE EXECUTABLE
IMPORTS
LINK BETWEEN THE EXECUTABLE AND WINDOWS LIBRARIES
DATA
INFORMATION USED BY THE CODE
e_magic 'MZ' CONSTANT SIGNATURE
e_lfanew 0x40 OFFSET OF THE PE HEADER 1
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/..
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
ImportsVA 0x2000 RVA OF THE IMPORTS 4
SECTIONS TABLE
NAME VIRTUALSIZE VIRTUALADDRESS SIZEOFRAWDATA POINTERTORAWDATA 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.
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);
IMPORTS STRUCTURES
DESCRIPTORS
0x203c 0x204c, 0 INT
0x2078 kernel32.dll 0,ExitProcess HINT/NAME
0x2068 0x204c, 0 INT
0x2044 0x205a, 0 INT
0x2085 user32.dll 0,MessageBoxA HINT/NAME
0x2070 0x205a, 0 IAT
ALL ADDRESSES HERE ARE RVA
CONSEQUENCES
AFTER LOADING,
0X402068 WILL POINT TO KERNEL32.DLL'S EXITPROCESS
0X402070 WILL POINT TO USER32.DLL'S MESSAGEBOXA
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
IAT
Hint,"API name"
library.dll
API_Address:
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!
OK
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
PE File Format

PDF 101 an Adobe document walkthrough

ANGE ALBERTINI
CORKAMI.COM



BASICS

PDF IS TEXT BASED, WITH BINARY STREAMS

TYPES

0: 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
[...]
3 0 obj
value
endobj
```

(OBJECT NUMBER) (REVISION NUMBER) R

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>
```

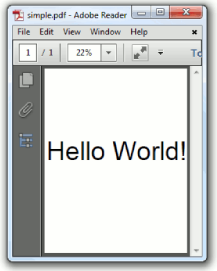
FOLLOWED BY XREF ENTRIES:
IF (OBJECT IN USE)
<OFFSET:10> <GENERATION:5> n
ELSE
<NEXT_FREE_OBJECT:10> <GENERATION:5> f

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

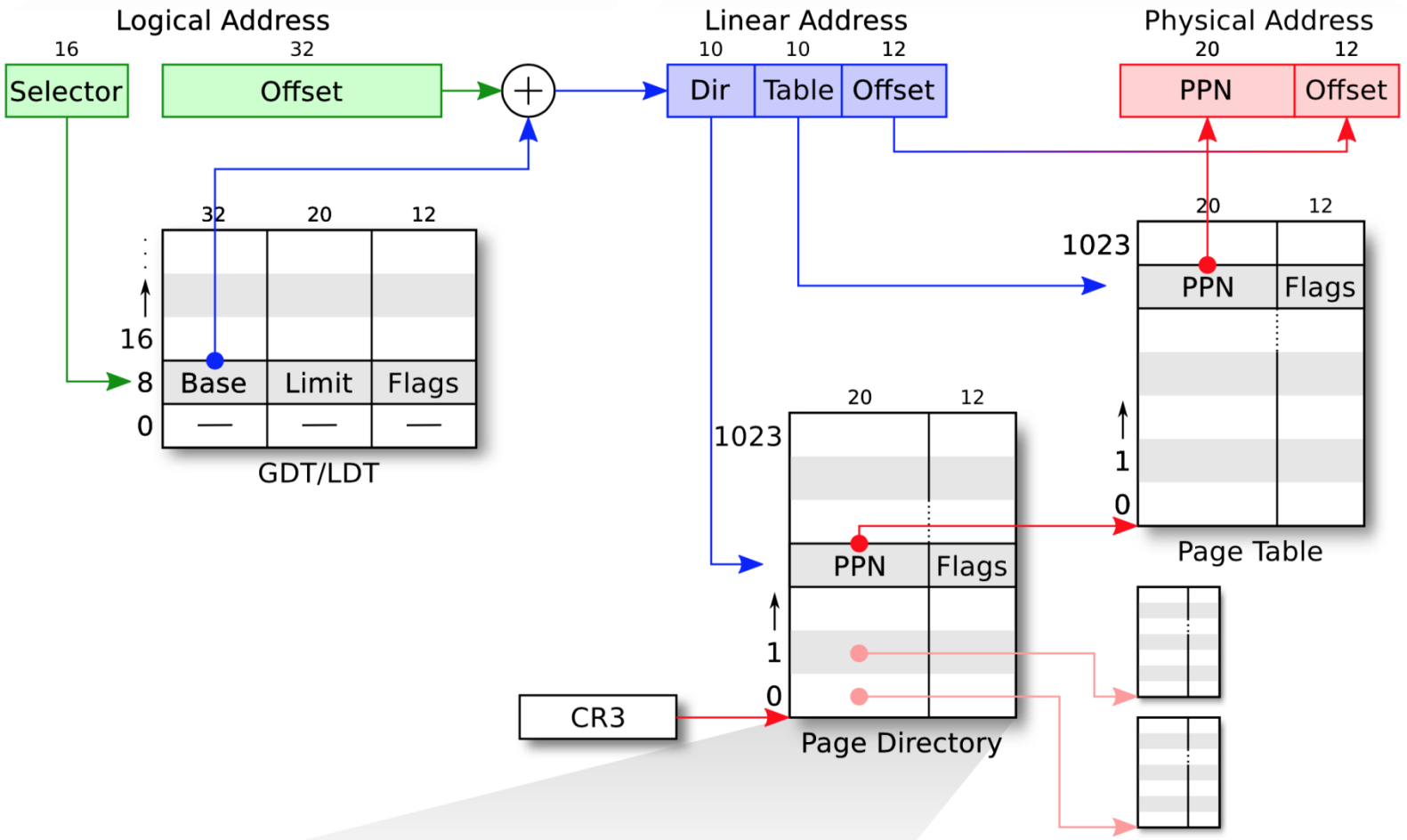
MEMORY ANALYSIS...

'cause reverse engineering ninjas are busy

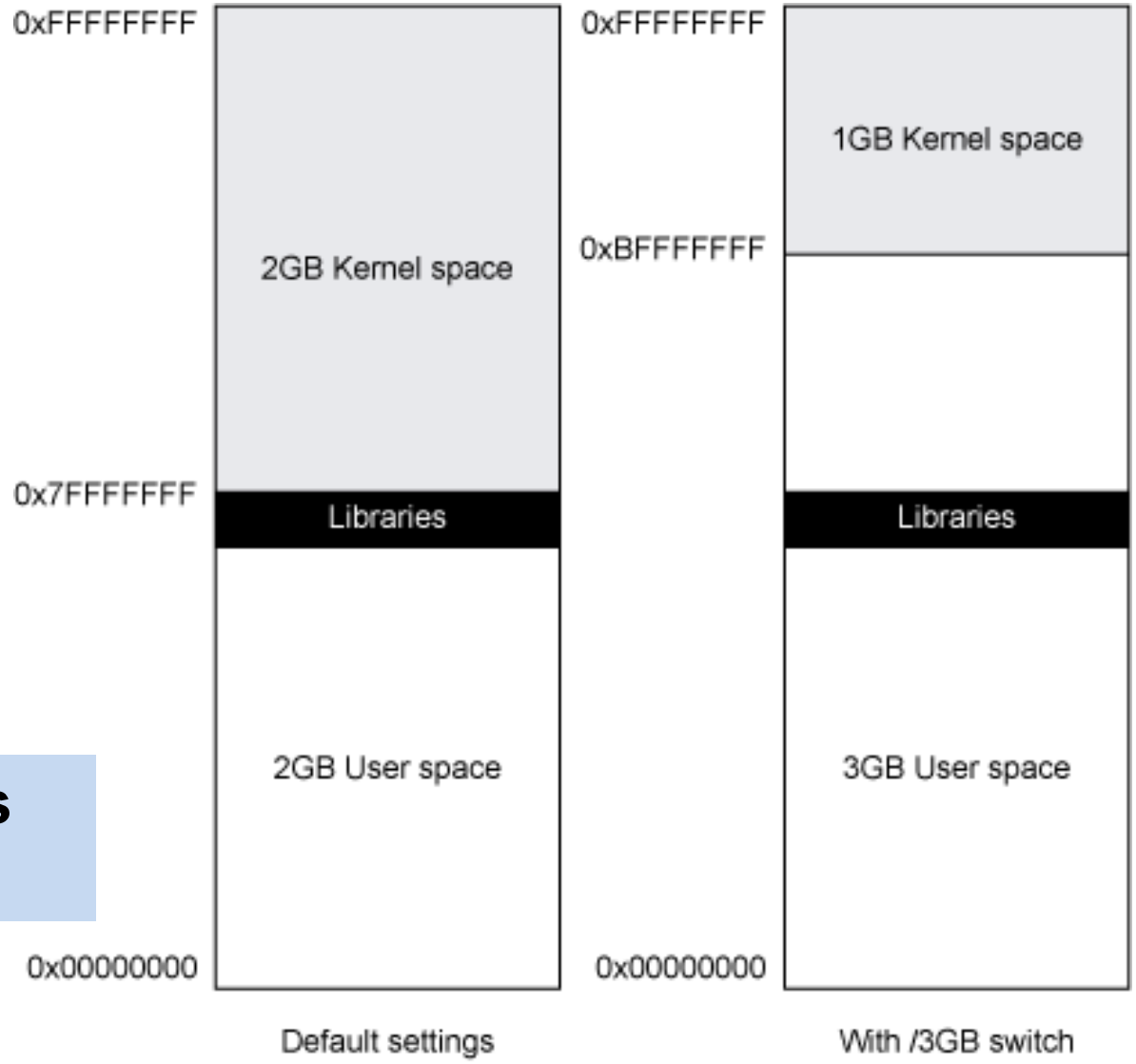
x86/x64 Memory organization

- 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

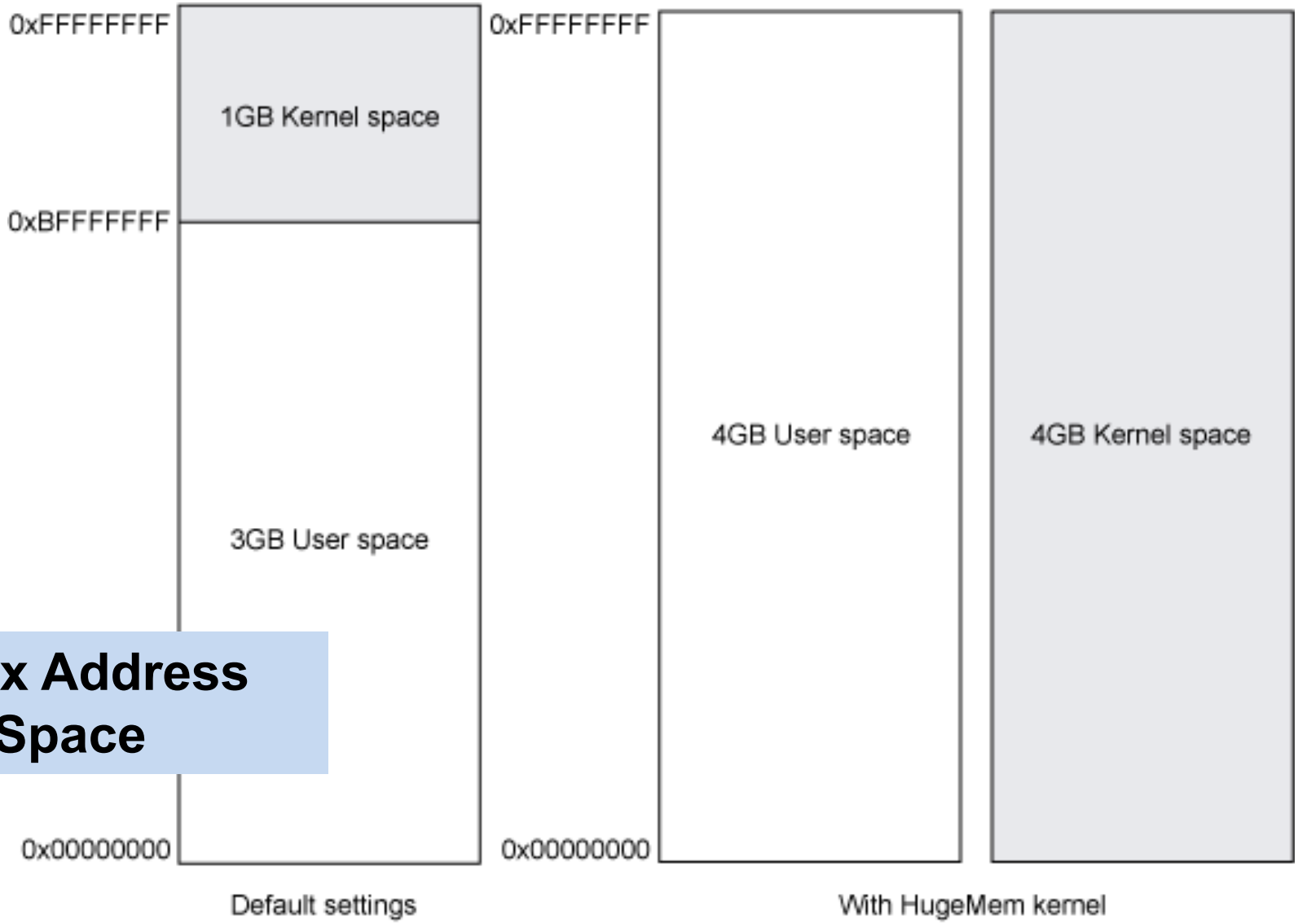
Segmentation | **Paging** | **Physical Address**

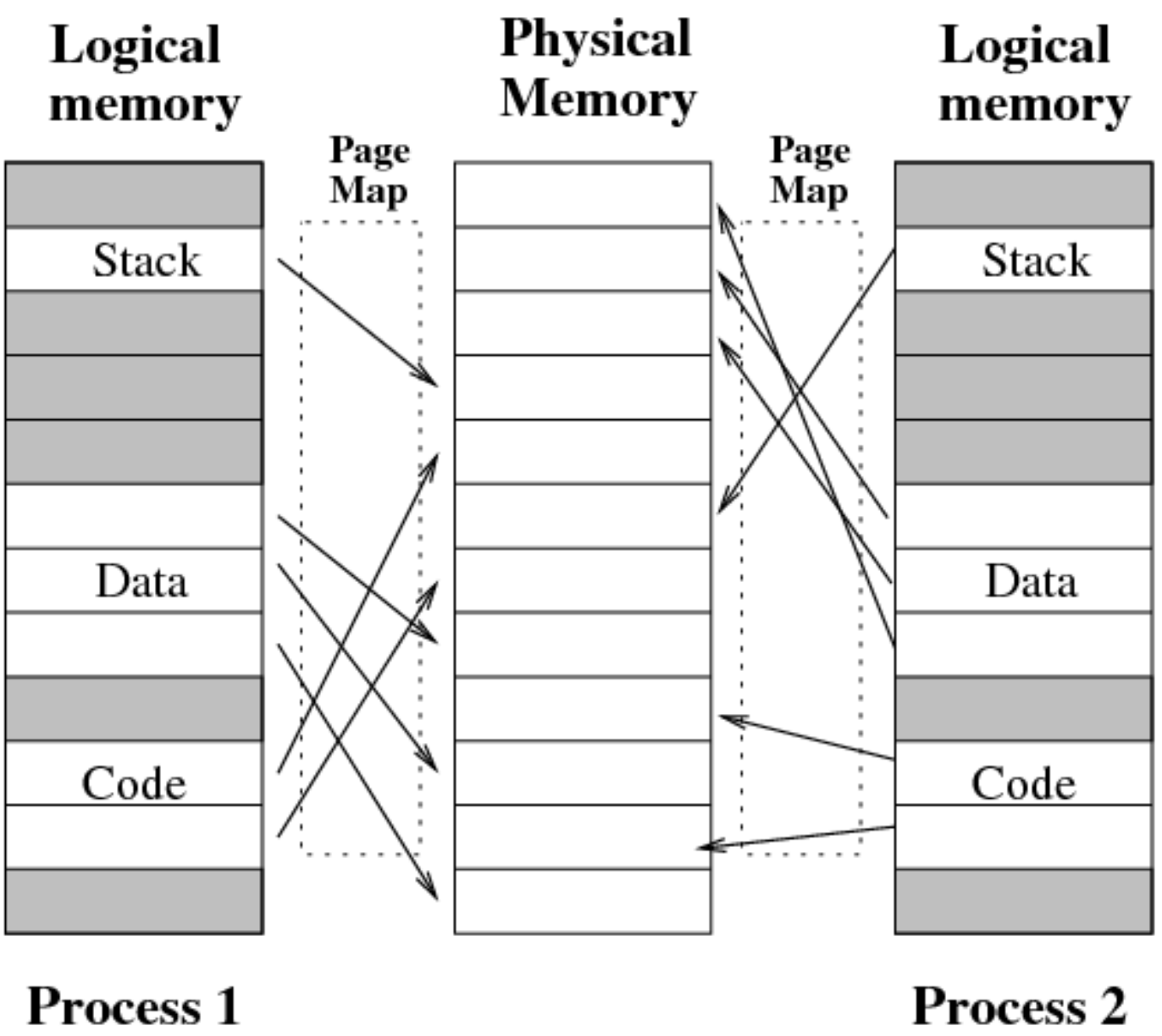


Win32 Address Space



Linux Address Space



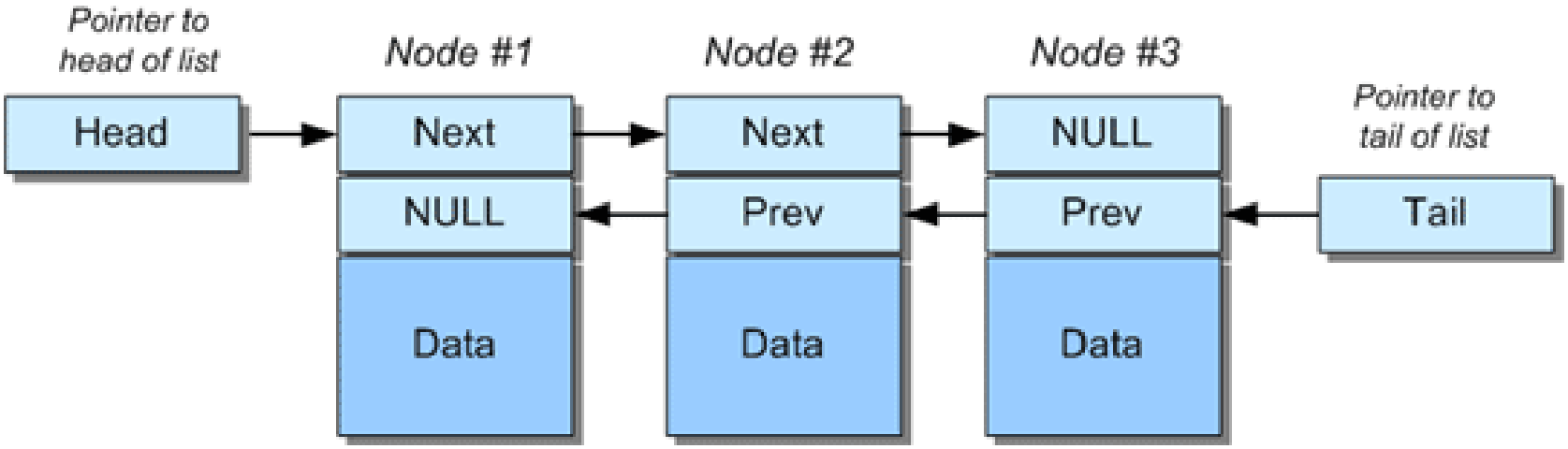

Logical memory
Physical Memory
Logical memory
Page Map
Page Map
Stack
Data
Code
Stack
Data
Code
Process 1
Process 2


MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?
0x3A28213A
0x6339392C,
0x7363682E.
I HATE YOU.

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

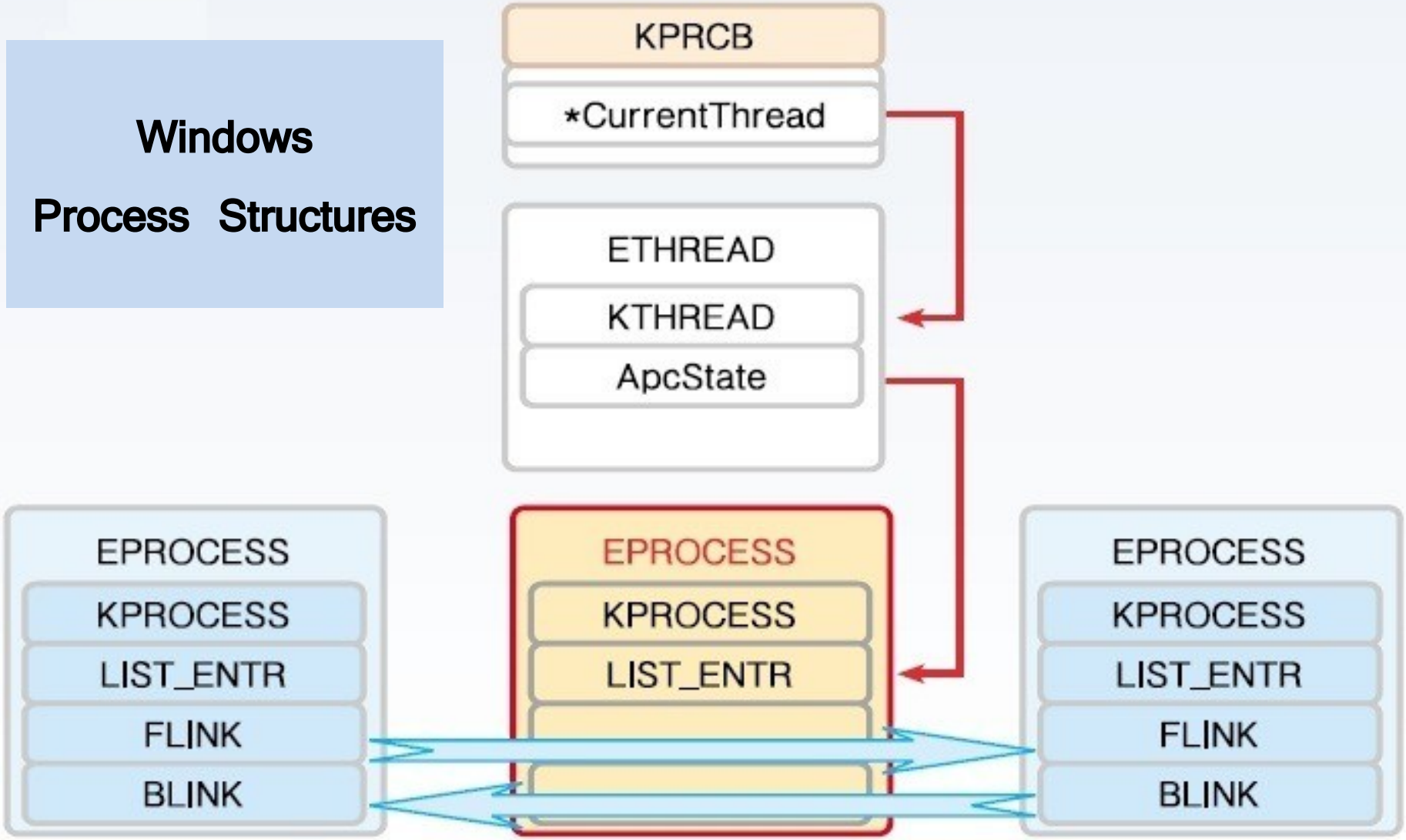
Double Linked Lists



DKOM – Direct Kernel Object Manipulation

- Dozens of various (double-)linked lists in Windows
  - 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

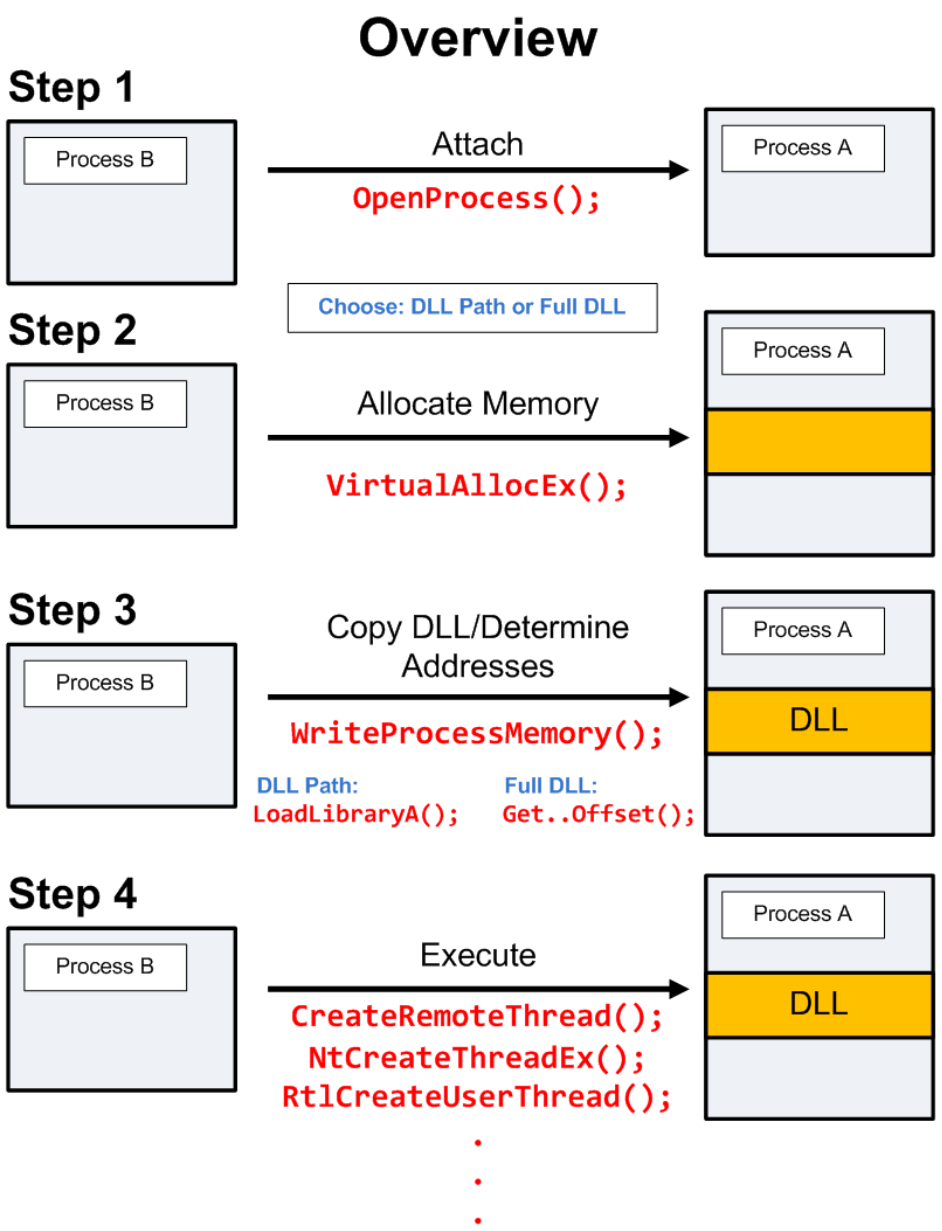
- Suspicious Memory Pages
- Processes
- Threads
- Sockets （Connections）
- Handles （Files）
- Modules/Libraries
- Mutexes
- LSA （Local Security Authority）
- Registry
- …

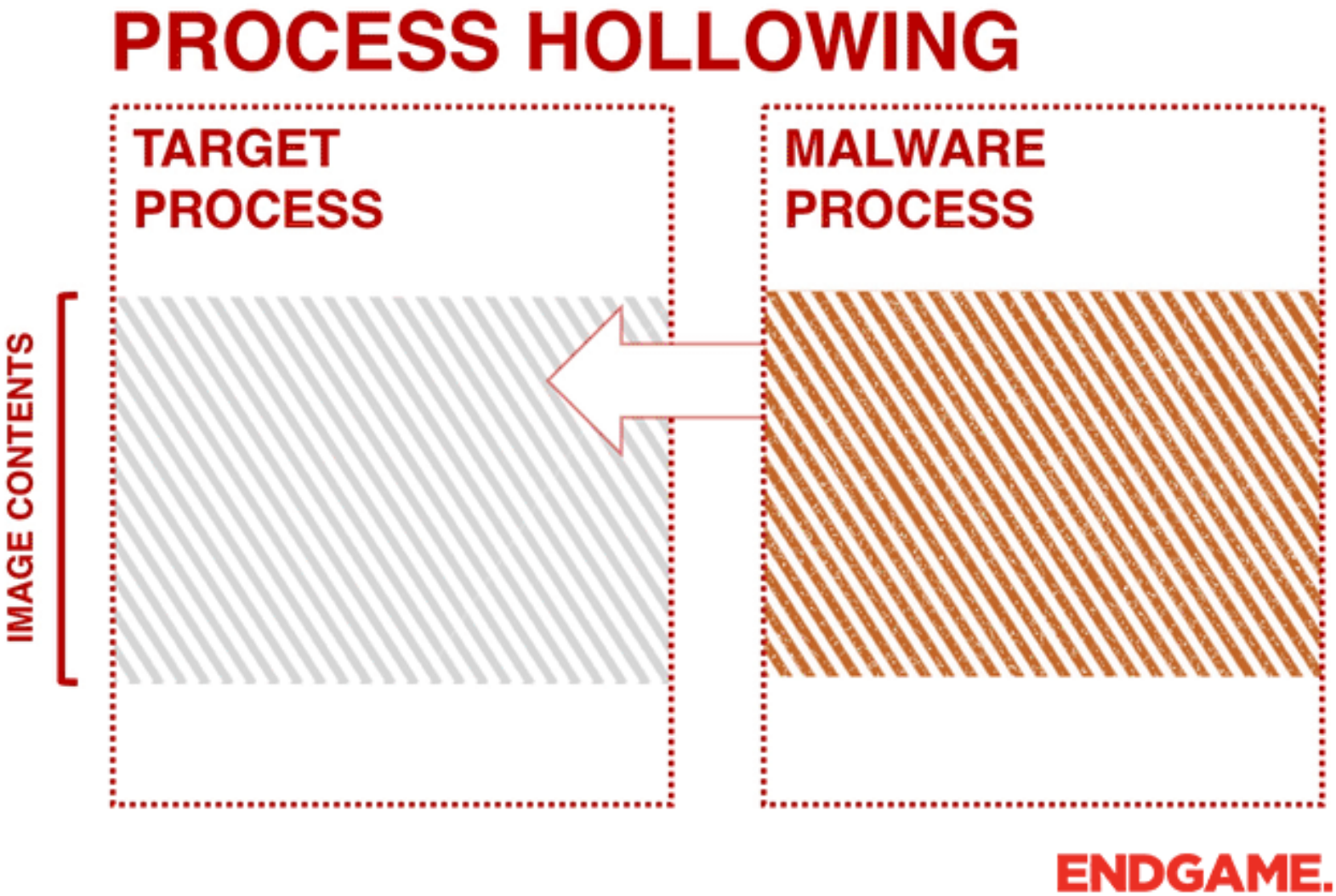
Memory Pages

- 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

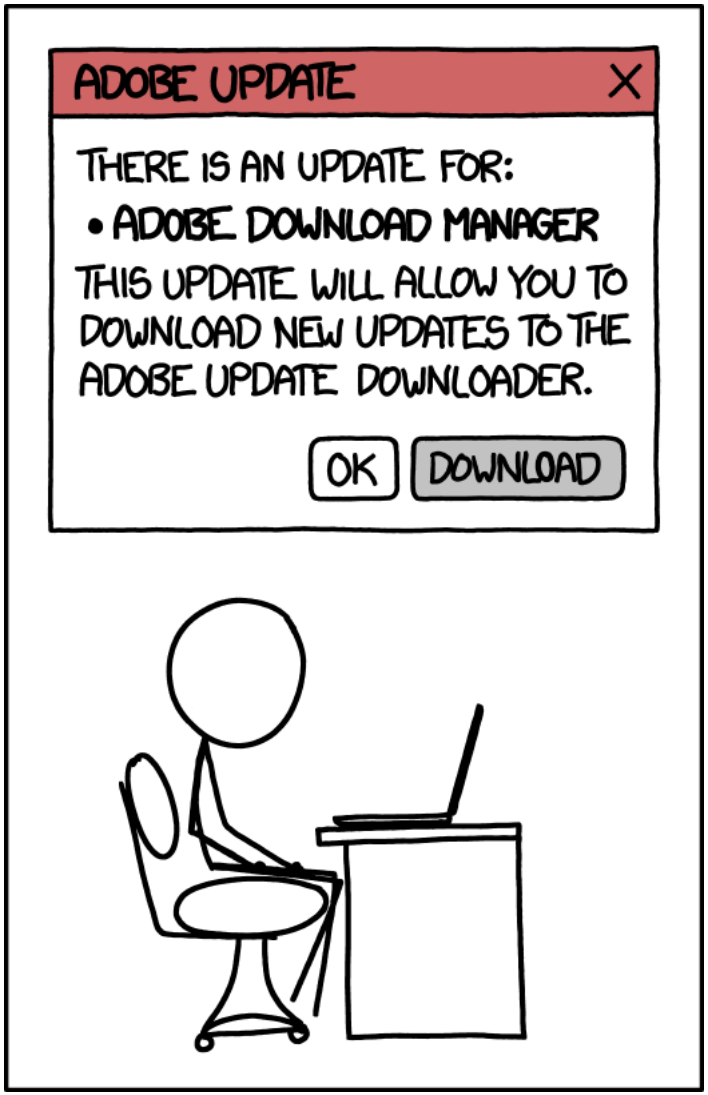
So that Internet Explorer behaves like a malicious process…




PROCESS HOLLOWING
TARGET PROCESS
MALWARE PROCESS
IMAGE CONTENTS
ENDGAME.

And now something completely…

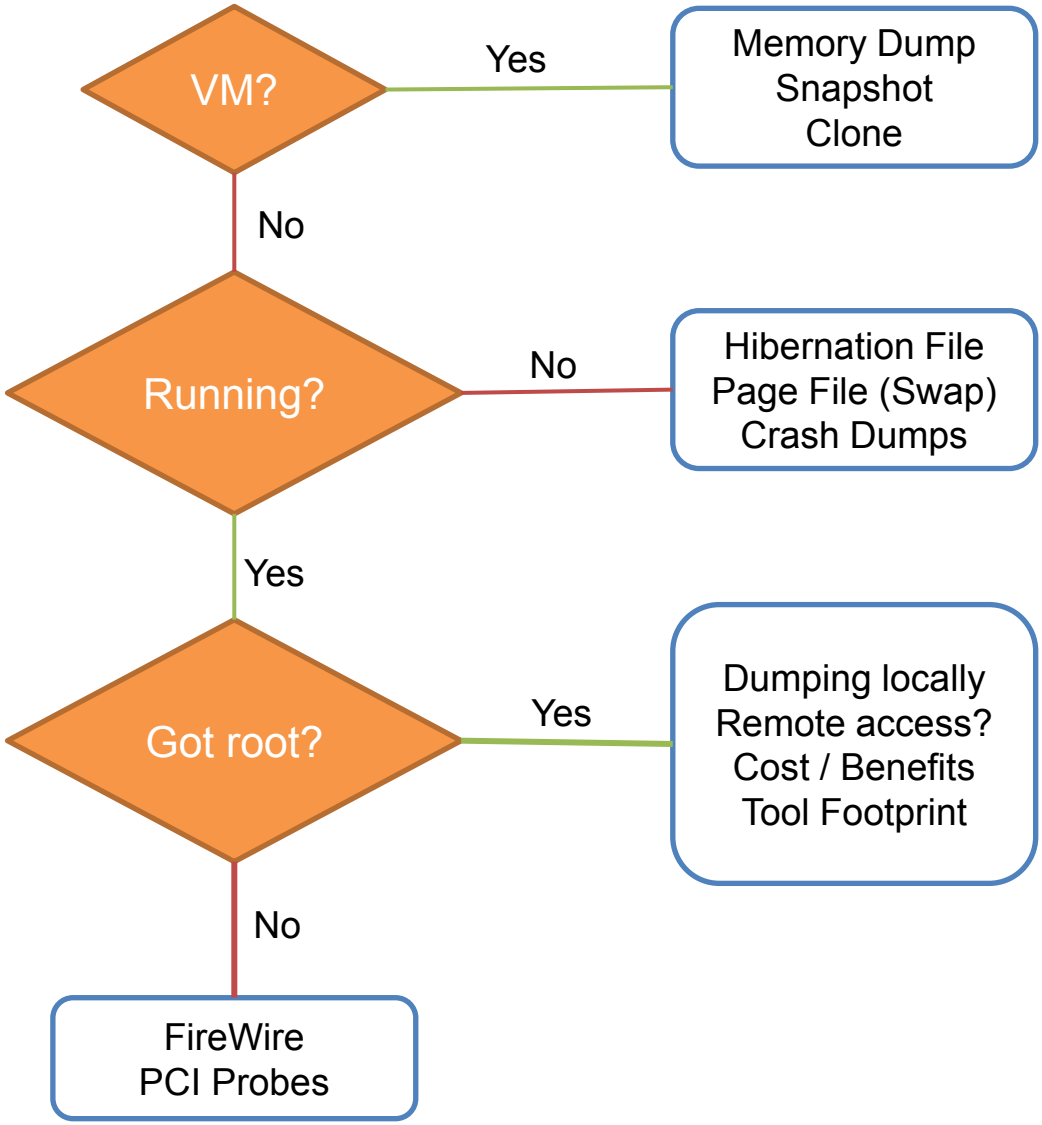
PRACTICAL


ADOBE UPDATE
THERE IS AN UPDATE FOR:
• ADOBE DOWNLOAD MANAGER
THIS UPDATE WILL ALLOW YOU TO DOWNLOAD NEW UPDATES TO THE ADOBE UPDATE DOWNLOADER.
OK
DOWNLOAD

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
  - Limited analysis options

Memory Acquisition



Memory Acquisition

- **Virtual Machines**
  - VMWare, VirtualBox, …
  - `VirtualBox -dbg -startvm "MalwareVM"` (and `.pgmphystofile` command or `vboxmanage debugvm`)
- Directly from the system! (if we have permissions to do that)
  - `windd`, `fastdump`, dumpit, `memorize`, **`winpmem`**
  - 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
    - However: malware can check for the available memory

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
  - Rekall (still a beta)

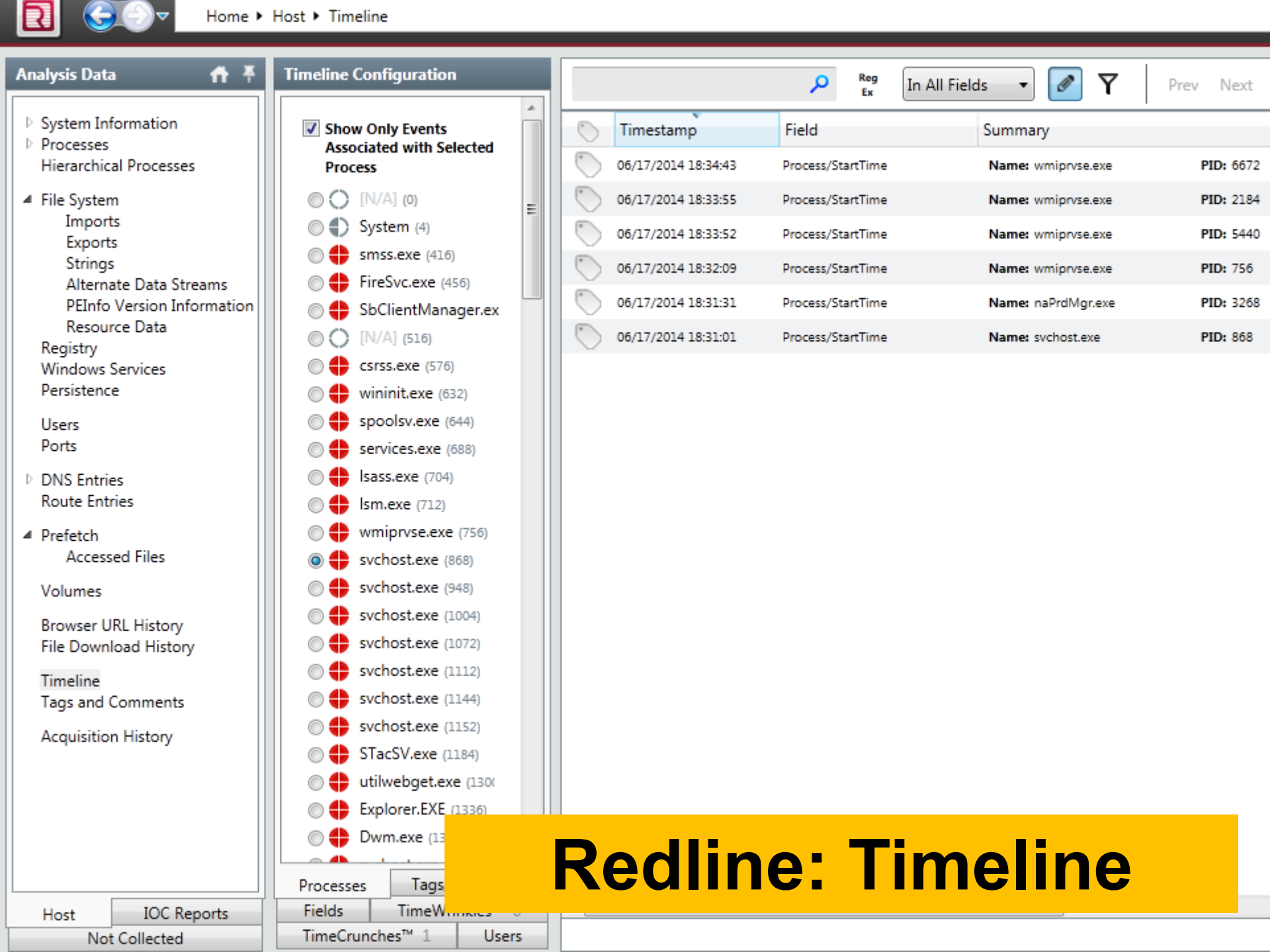
Memory Analysis Tools

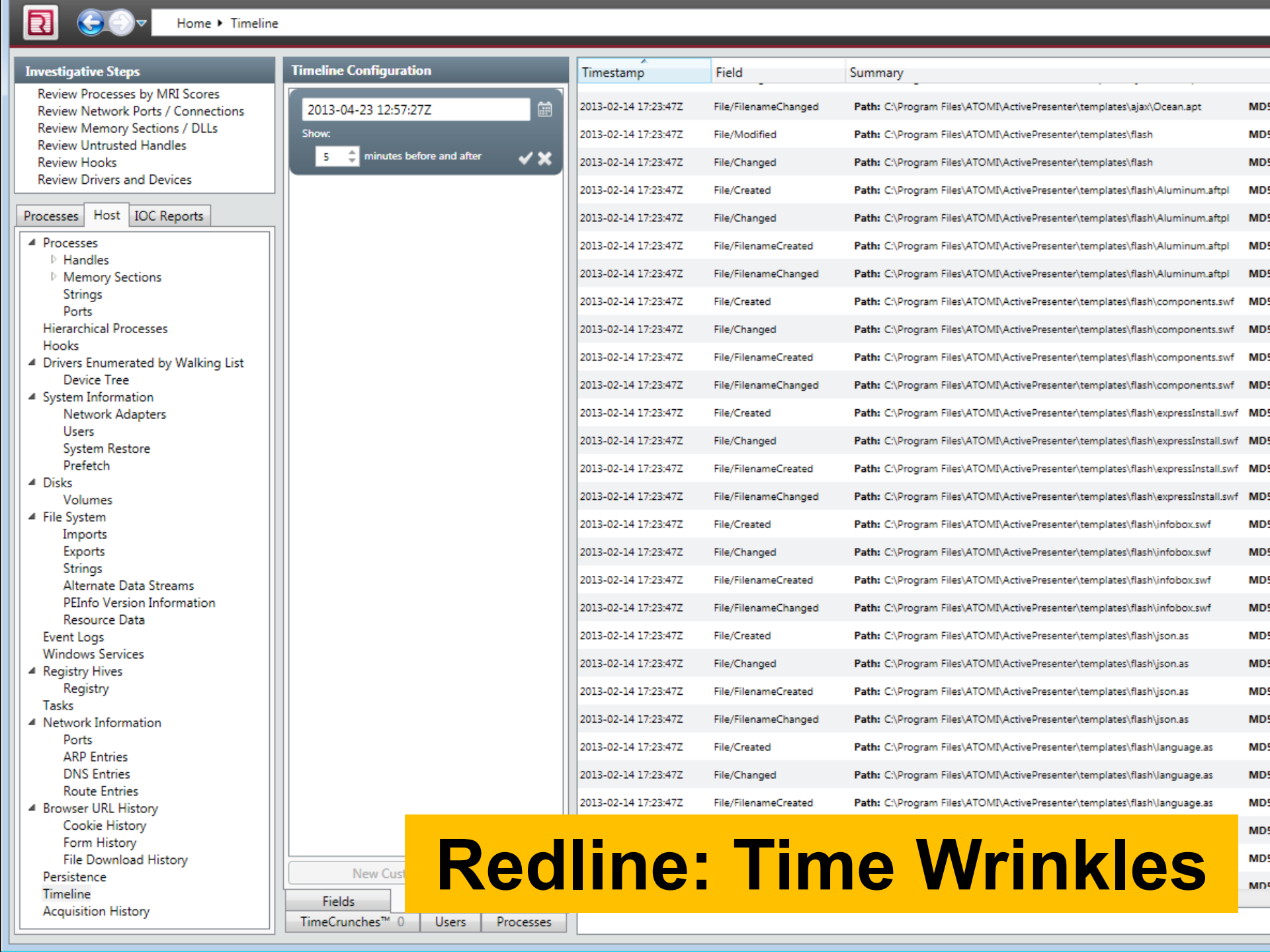
- Mandiant Redline
  - Free, available for Windows
- HBGary/CounterTack Responder (CE/Pro)
  - Community Edition available against registration
- **Volatility Framework**
  - Open source, no GUI
- Google Rekall
  - Open source, 'Volatility done right', GUI
  - Google supported (part of GRR agent)

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®
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: Start


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 (13
Processes
Tags
Fields
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: Timeline


Home ▸ Timeline
Investigative Steps
Review Processes by MRI Scores
Review Network Ports / Connections
Review Memory Sections / DLLs
Review Untrusted Handles
Review Hooks
Review Drivers and Devices
Processes
Host
IOC Reports
Processes
Handles
Memory Sections
Strings
Ports
Hierarchical Processes
Hooks
Drivers Enumerated by Walking List
Device Tree
System Information
Network Adapters
Users
System Restore
Prefetch
Disks
Volumes
File System
Imports
Exports
Strings
Alternate Data Streams
PEInfo Version Information
Resource Data
Event Logs
Windows Services
Registry Hives
Registry
Tasks
Network Information
Ports
ARP Entries
DNS Entries
Route Entries
Browser URL History
Cookie History
Form History
File Download History
Persistence
Timeline
Acquisition History
Timeline Configuration
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
2013-02-14 17:23:47Z File/Modified Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl
2013-02-14 17:23:47Z File/FilenameChanged Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\Aluminum.aftpl
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf
2013-02-14 17:23:47Z File/FilenameChanged Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\components.swf
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf
2013-02-14 17:23:47Z File/FilenameChanged Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\expressInstall.swf
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf
2013-02-14 17:23:47Z File/FilenameChanged Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\infobox.swf
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as
2013-02-14 17:23:47Z File/FilenameChanged Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\json.as
2013-02-14 17:23:47Z File/Created Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as
2013-02-14 17:23:47Z File/Changed Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as
2013-02-14 17:23:47Z File/FilenameCreated Path: C:\Program Files\ATOMI\ActivePresenter\templates\flash\language.as
Fields
TimeCrunches™ 0
Users
Processes
Redline: Time Wrinkles

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
- Replaced with CounterTack Responder Pro

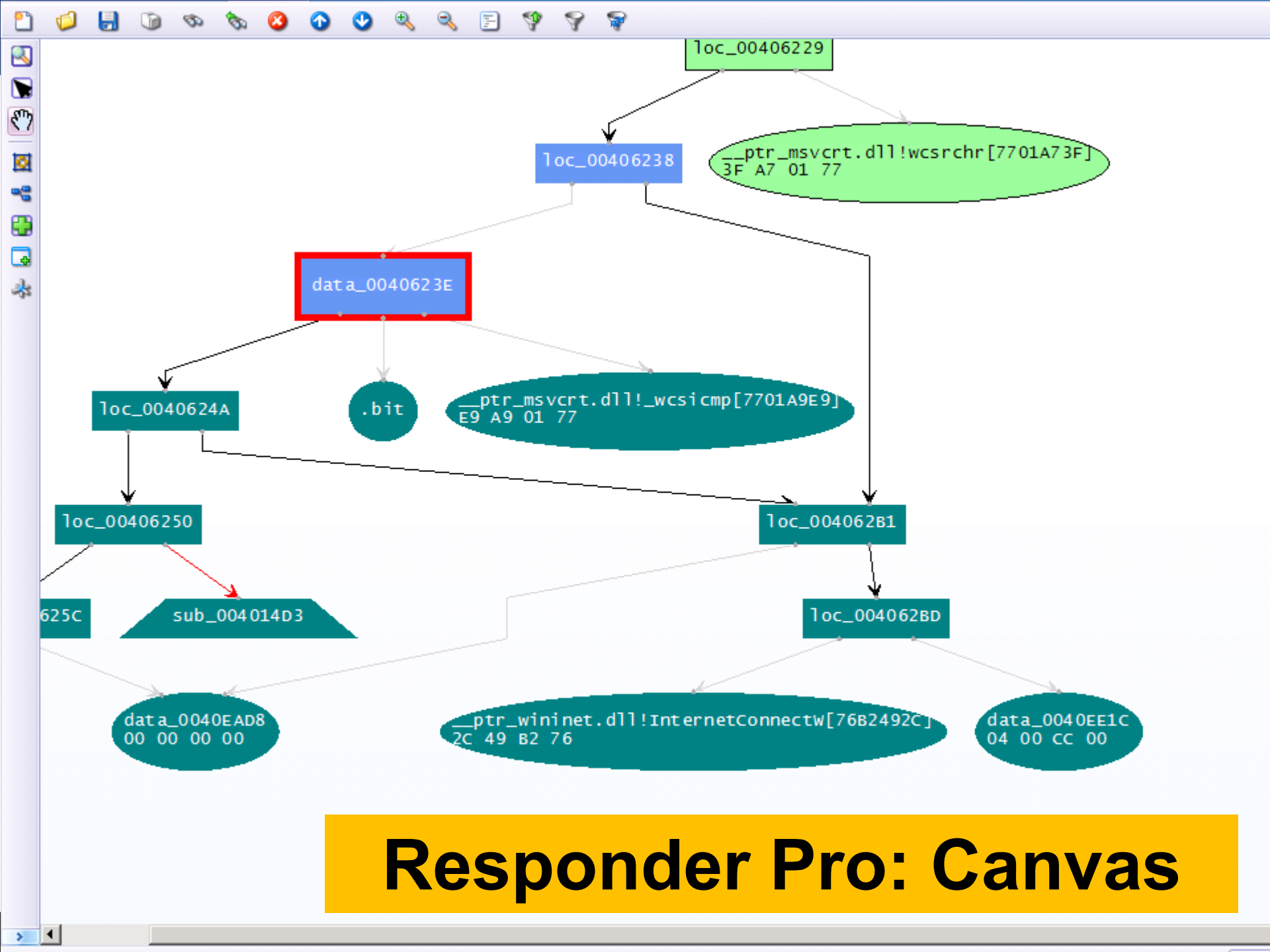
HBGary Responder Pro -- DDNA

- 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?

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		
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
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
suspicious.
Responder Pro: DDNA



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 \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 acquired executables to VirusTotal
    - MD5/SHA1 of the dump 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)
  - Malware Traffic Analysis (both traffic & 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)
  - Malware Traffic Analysis (both traffic & samples)

Thank you for your attention.

Answers & Questions